**UNIVERSITY OF OSLO**
**Department of Informatics**

# Privacy by Design: Adding Privacy to Publishing Platforms

Master Thesis

Joakim Aleksander Olsen Valla

**May 2013**

**Acknowledgments**

First of all, I wish to thank my supervisor Gisle Hannemyr, for his guidance throughout the work on this thesis.

I would also like to thank those who where willing to participate in the study conducted in the work with this thesis.

I am also grateful to my friends for providing encouragment when I needed it.

Finally, I would like to thank my family for being so supportive and for always being there for me.

**Abstract**

Privacy can be considered a measure of degree of control over personal data. While information system owners have control over personal data as long as it is stored on a system under their immediate control, this control diminishes when data is copied from the information system (e.g., a website) and to a different system. This happens today to personal data that is crawled by a web robot and all the information it exposes is copied into a database belonging to a web search engine provider. The web is full of personal names, which is usually attached to some contextual data. If these personal names are indexed by web search engines, along with the contextual data attached to them, both will be discoverable by anyone searching for a specific name. While some such discoveries may be beneficial to the subject, others may be harmful. The aim of this thesis is to promote the need for better methods of controlling personal data published online. Specifically, the focus is on methods for shielding personal names from web search engines. Popular Content Management Systems, like WordPress, does very little to help publishers hide personal data from search engines. This thesis conducts an empirical study on the Robots Exclusion Protocol, investigating its effectiveness as a method for controlling indexing and crawling by search engines. Our study, which is limited to the Google search engine, suggests that, as long as the different directives are used correctly, the protocol can be considered quite reliable as a method for preventing content from being indexed by Google. As a response to the privacy challenges related to personal names being discoverable through web search engines, as well as the lack of fine-grained control offered by the Robots Exclusion Protocol, we have proposed and implemented a solution for increasing control over personal data published online. It accomplishes this by preventing search engines from indexing personal names along with contextual data attached to those names. By allowing the user to specify that specific parts of the content is to be kept out of the index, the tool offers publishers more fine-grained control over their data than the Robots Exclusion Protocol. The solution has been implemented as a plugin for the WordPress publishing platform. It has been installed on an experimental website, and we have verified that while the rest of the content has been indexed and is discoverable through the Google search engine, personal names posted on the website are no longer discoverable.

***Keywords***: privacy, robots, internet, search engines, crawling, indexing, robot exclusion protocol, wordpress, plugin

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Search engine providers play a crucial role in the information society as intermediaries, by crawling and indexing the huge amounts of content available on the World Wide Web, and then providing an easy and effective way for people to find that information. However, search engines are also the cause of several challenges to privacy.

Over the years there have been several incidents where a human or technical error has led to sensitive personal data being made publicly available online. In many of these cases, the situation has been made even worse when the files have been indexed by Google and other web search engines. In an article dated 29. September 2009, VG Nett [1] describes how a hospital published lists containing the names of people who had been infected with the swine flu, online. Despite removing the information from their website once they became aware of their mistake, the information was still available through search engines, due to the fact that a copy of the files had been stored in the search engines' so-called "cache". Another example is described in an article by Digi.no [2], dated 29. June 2011, involving the discount service Groupon's Indian subsidiary Sosasta.com. In this case, their entire database containing the user information of 300.000 customers was published online. The problem was made even worse by the fact that the information was then indexed by search engines like Google. The person who discovered the leak, came across the SQL file containing the information by using Google to search for the relevant file type in combination with expressions like "password" and "gmail".

Then there is the issue of how the Internet "never" forgets something. Once something has been published on the Internet, it can be almost impossible to remove it again. You first need the cooperation of the administrator of the website where it has been published and have them remove it. If the content has already been crawled by a search engine, you then need to have the content removed from there, which is not always so easy. Some search engines, like Google, cache their results, enabling surfers to view content

after it has been removed from the original website until the search engine once again crawls the website in question. In addition, the Internet Archive's Wayback Machine[1] stores records of websites dating back to the 1990s. And while their website explains how people can avoid having their site's pages excluded from the Wayback Machine, other sites are not always so helpful.

While search engines play a vital role in providing everyone with an easy way of looking up all kinds of information, it comes at the price of individuals, or "search targets", whose private lives and potentially personal information becomes easily accessible and searchable by anyone. The web is full of personal names, which is usually attached to some contextual data, like utterances or images. If these personal names are indexed by search engines, along with the contextual data attached to them, both will be discoverable by anyone searching for a specific name. While some such discoveries may be beneficial to the subject, others may be harmful. For example, it is becoming more and more popular for potential employers to google new applicants before being called in for a job interview, and what they find may have an effect on whether the applicant gets the job or not. A study conducted by Microsoft and Cross-Tab found that as much as 70 percent of U.S.recruiters and HR professionals say they have rejected candidates based on information they found online [3].

## 1.1 Research Focus

The overall motivation for this thesis is the need for better methods of controlling personal data that is published online. Specifically, the focus is on methods for shielding personal names from search engines. According to a report presented by the Privacy Commission in 2009 [4], the discourse around the concept of privacy has always been characterized by terminological ambiguity, confusion and controversy. In order to say something about how to improve the privacy aspect of a design we first need to have a clear understanding of the concepts involved. The first objective of our research will therefore be to:

- ***Clarify* the concepts of privacy and data protection.**

While information system owners have control over personal data as long as it is stored on a system under their direct control, this control diminishes if that data is copied into a database belonging to a search engine provider, such as Google. This can happen if the personal data is exposed through some sort of web Content Management System (CMS), and that CMS is crawled by a search engine robot. We have already seen some examples of how search engines can increase the damage potential in situations where sensitive personal data is made publicly available online by a mistake. This

[1] `http://archive.org/web/web.php`

makes it important to know how personal data can be protected from search engines. Our second objective will therefore be to:

- ***Investigate* how online publishers can hide personal data from search engines.**

Armed with the knowledge of how information published online can be indexed by web search engines, and what steps online publishers can take in order avoid personal data being indexed, we will begin the process of developing a tool for increasing control over personal data that is published online. Our third objective will therefore be to:

- ***Develop* a tool for increasing control over personal data that is published online.**

It is not sufficient to just build a solution, we also have to evaluate its effectiveness and prove that the solution actually functions as intended. Our fourth and last objective will therefore be to:

- ***Evaluate* the tool's ability to facilitate increased control over personal data published online.**

The main goal of this dissertation is to develop a tool for facilitating increased control over personal data published online. More specifically, the tool will allow publishers to redact personal names from content that is viewed by search engine robots, with the aim of avoiding personal names being indexed along with any contextual data attached to them.

## 1.2 Contributions

As part of our investigation on how online publishers can hide personal data from search engines, we have conducted an empirical study on the *Robots Exclusion Protocol* (REP). Despite its importance, there seems to be relatively few studies investigating its usage in detail. Two such studies, mentioned in Chapter 4, investigated how a number of websites had deployed the REP, and revealed many incorrect usages. And while the number of studies investigating the usage of the protocol are relatively few, there seems to be no studies conducted that investigates its effectiveness as a method for controlling indexing and crawling by search engines. Our study, which is limited to the Google search engine, suggests that, as long as the different directives are used correctly, the REP can be considered quite reliable as a method for preventing content from being indexed by Google.

As a response to the privacy challenges related to personal names being discoverable through web search engines, as well as the lack of fine-grained control offered by the REP, we have proposed and implemented a solution for increasing control over personal data published online. It accomplishes

this by preventing search engines from indexing personal names along with contextual data attached to those names. By allowing the user to specify that specific parts of the content is to be kept out of the index, the tool offers publishers more fine-grained control over their data than the REP. The solution has been installed on an experimental website, and we have verified that while the rest of the content has been indexed and is discoverable through the Google search engine, personal names posted on the website are no longer discoverable.

## 1.3 Chapter Overview

In Chapter 2 we delve more deeply into the field of privacy and data protection. There we explain some of the relevant concepts, before providing an overview of the most important laws and regulations governing privacy and data protection, focusing on Norway and the European Union. We also provide a description of two important concepts related to privacy in computing applications: *Privacy Enhancing Technologies* and *Privacy by Design*. As this thesis will be concerned with how to protect personal data from web search engines, this chapter will also clarify how search engines work and operate. In Chapter 3 we describe our choice of research methods employed in this thesis. Chapter 4 provides an in-depth study of the *Robots Exclusion Protocol*, which is a method website owners can use to control how robots, and specifically search engine robots, crawl and index content on the web. Following this, in Chapter 5 we investigate to what degree the publishing platform WordPress allows publishers to hide personal data from search engines. In Chapter 6 we present our solution for controlling personal data published online. This solution is then evaluated in Chapter 7. In Chapter 8 we summarize and conclude this thesis.

# Chapter 2

# Background

In this chapter we briefly explore the subject of privacy and data protection. Then, in Section 2.2, we take a look at two concepts that are often referred to in the context of privacy in computing applications - Privacy by Design and Privacy Enhancing Technologies. Finally, in Section 2.3 we take a closer look at how web search engines functions and operates.

## 2.1 Privacy and Data Protection

There is a vast body of literature on the subject of privacy and the protection of data, and it is well outside the scope of this thesis to try to summarize it all. Instead, Section 2.1.1 will first attempt to give a short explanation of the concept of privacy. Suggested definitions for the terms "privacy" and "data protection" will be presented. Section 2.1.2 on page 7 will then give a brief overview of the most important laws and regulations governing privacy and data protection, focusing on Norway and the European Union.

### 2.1.1 The Concept of Privacy

The term "privacy" is frequently used in discussions ranging from philosophical, political, and legal, as well as in ordinary language. Yet there is no single definition or meaning of the term. According to a report presented by the Privacy Commission in 2009 [4], the discourse around the concept of privacy has always been characterized by terminological ambiguity, confusion and controversy. In "Privacy and Data Protection in an International Perspective", Bygrave [5] identifies four principal ways of defining privacy that the privacy debate has revealed: definitions in terms of *non-interference*; definitions in terms of *limited accessibility*; definitions of privacy being conceived as *information control*; and definitions incorporating various elements of the other three sets of definitions, but where privacy is linked exclusively to *intimate* or *sensitive* aspects of persons' lives.

While privacy as a concept has existed for a long time, more systematic written discussion of the concept of privacy is, according to [6], often said to begin with an article in the Harvard Law Review by Samuel Warren and Louis Brandeis, titled "The Right to Privacy" [7]. The article focused in large part on the practices of the press and the publicity allowed by the inventions of photography and newspapers. Since then, there have been numerous debates in the United States related to privacy, ranging from abortion-issues to the practice of wiretapping. In the 1960s, the discussion turned to the implications of computerized processing of personal data. After the Second World War, an increasing number of federal agencies gathered huge amounts of detailed information on American citizens and residents, and with the advent of computer technology it became easier to cross-reference individuals' personal data. After attempts to create a national data center that was supposed to collect data from all the federal agencies for statistical purposes, and in response to concerns about how computerized databases might impact on individuals' privacy, the American Congress passed the Privacy Act in 1974. The Privacy Act required all federal agencies to abide by certain principles related to modern data protection, but was in many ways flawed, mainly due to a lack of enforcement.

The debate in the US in the 1960s and 1970s concerning the privacy-related threats posed by modern information and communications technologies (ICT) exercised, according to Bygrave [5], a considerable influence on related debates in other countries. In Europe, however, such debates preferred to use the term "data protection", but, as Bygrave points out in [5], these terms are not completely synonymous, at least from a European perspective. This is further reflected in the fact that data protection is increasingly being treated in European law as a separate set of rights from the more traditional right to respect for ones privacy or private life.

There is no direct translation of the English word "privacy" to the Norwegian language. In Norway, "personvern" ("protection of person(ality)") has traditionally been the preferred term when discussing privacy, but there has been much debate as to the definition of the term, and what it encompasses. In 2007 the Norwegian Government appointed a Privacy Commission. Its remit included a comprehensive stock of the challenges to privacy, identifying and evaluating the policy instruments that currently exists to preserve privacy, and promoting proposals for new principles and instruments [8]. In its final report [4], the Privacy Commission suggested, among other things, that a clear distinction is made between the terms "personvern" (eng: privacy) and "personopplysningsvern" (eng: data protection). They defined the two terms as follows:

- "Personvern dreier seg om ivaretakelse av personlig integritet; ivaretakelse av enkeltindividers mulighet for privatliv, selvbestemmelse (autonomi) og selvutfoldelse. Eksempel på en person-

> vernbestemmelse er vernet av privatlivets fred i straffeloven § 390.
>
> - Person*opplysnings*vern dreier seg om regler og standarder for behandling av personopplysninger som har ivaretakelse av personvern som hovedmål. Reglenes formål er å sikre enkeltindivider oversikt og kontroll over behandling av opplysninger om dem selv. Med visse unntak skal enkeltpersoner ha mulighet til å bestemme hva andre skal få vite om hans/hennes personlige forhold. Det er denne delen av personvernretten som er underlagt den mest omfattende lovregulering i for eksempel personopplysningsloven, helseregisterloven, regler om taushetsplikt og så videre." [4, p. 32]

By making this distinction, the Privacy Commission attempted to bring the Norwegian legal terms in line with European legal theory, pointing to the fact that it is becoming more common within the European Union to make a legal distinction between the right to respect for ones private life, or privacy, on the one hand, and the right to protection of personal data on the other.

In this thesis, the terms "privacy" and "data protection" will be used according to the definitions of "personvern" and "personopplysningsvern" suggested by the Privacy Commission.

### 2.1.2 Laws and Regulations

The right to privacy is protected through several international instruments containing rules of various degrees of detail, scope and legal status. Some declares privacy a fundamental human right, and deals with the concept in very general terms. For example, the UN's Universal Declaration of Human Rights of 1948 states in Article 12 that:

> "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks."[9]

Likewise, the European Convention on Human Rights of 1950 declares in Article 8(1) that:

> "Everyone has the right to respect for his private and family life, his home and his correspondence." [10]

These and other treaties form the normative foundation for more detailed sets of rules containing specific requirements for the processing of personal data. Of these, the Data Commission points, in its report from 2009 [4], to the European Data Protection Directive of 1995 (EU Directive 95/46/EC) and

the Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data of 1981 as the two most important regulations.

The Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data is, according to [4], the only convention which directly relate to personal data protection. The Council of Europe's website declares it "the only binding international legal instrument in the field, with a potential worldwide scope of application" [11]. It has been ratified by most of the member countries of the European Council, including Norway. The purpose of the convention is (according to Article 1) to secure each individual his rights and freedoms, and in particular his right to privacy with regard to the automatic processing of personal data. [12]

In 1995, the EU adopted a directive on personal data protection. The purpose of EU Directive 95/46/EC is to protect individuals with regard to the processing of personal data and on the free movement of such data. In 1999, the Directive was included in the EEA Agreement, and was thus made legally binding for Norway. It is implemented in Norway mainly by the Personal Data Act of 2000 ("personopplysningsloven"), which is the main law regulating data protection in Norway. The purpose of the Personal Data Act (PDA) is to:

> "... protect natural persons from violation of their right to privacy through the processing of personal data. The Act shall help to ensure that personal data are processed in accordance with fundamental respect for the right to privacy, including the need to protect personal integrity and private life and ensure that personal data are of adequate quality" [13]

The PDA is governed and overseen by the Data Inspectorate ("Datatilsynet"), who's responsibilities include verifying that statutes and regulations which apply to the processing of personal data are complied with, and that errors or deficiencies are rectified; identify risks to protection of privacy, and provide advice on ways of avoiding or limiting such risks. [14]

### 2.1.3 Personal Data

In privacy law, different terms for personal data are being used, which may or may not be equivalent, depending on the jurisdiction, context, or purposes for which the term is being used. Common terms, beside personal data, are personal information and personally identifiable information (PII). The abbreviation PII has four common variants based on personal, personally, identifiable, and identifying [15]. So for a more precise definition of personal data, one has to consult the laws and regulations of each country or institution (e.g. the EU). Included below are two definitions of personal data. The first definition comes from EU Directive 95/46/EC, also known as the

"European Data Protection Directive", and the second is from the Norwegian Personal Data Act of 2000.

In Article 2 (a) of the European Data Protection Directive, definitions are given for *personal data* and *data subject* (for the purposes of the directive):

> "'personal data' shall mean any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity" [16]

The European Data Protection Directive is intended, according to Article 1 (1), to protect the fundamental rights and freedoms of natural persons, and in particular their right to privacy, with respect to the processing of personal data. In addition to establishing several conditions that must be met in order to process personal data legitimately, the Directive (in Article 8 (1)) places heightened restrictions on the processing of what it calls special categories of data, which consists of "personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, and ... data concerning health or sex life" [16], stating that member states shall prohibit the processing of such data (unless certain exceptions apply). Article 2 (b) specifies that, for the purposes of the directive, processing of personal data means:

> "... any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, organization, storage, adaption or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction" [16]

In Norway, the European Data Protection Directive is implemented as "Personopplysningsloven" (Personal Data Act), and is the main law regulating data protection in Norway. According to the Norwegian Personal Data Act, personal data is (for the purpose and scope of the act):

> "... any information and assessments that may be linked to a natural person" [13].

This is a derivation of the EU Directive, and is considered equivalent to it. Although the criteria that data must be related to an identified or identifiable natural person mentioned in the EU Directive is not clearly stated in the Norwegian legal text, according to [4] it follows from the legislative history that the same must also be deployed in accordance with Norwegian law.

Both of these directives defines the term "personal data" quite broadly, as well as what it means to process such data. In 2003, the European Court of

Justice (ECJ) handed down a ruling which clarified the application and scope of the European Data Protection Directive. Case C-101/01 [17] arose after Bodil Lindqvist, a Swedish woman who worked as a catechist at a Swedish church, posted some information about her colleagues on her own website (what we today would usually call a "blog"). The information in question included names, phone numbers, and hobbies, as well as some health-related information (she at one point stated that one colleague had injured her foot and was on half-time on medical grounds). At no point had Mrs Lindqvist informed her colleagues about the existence of those pages, nor had she notified the Swedish data protection authorities ("Datainspektionen") of her activity. Despite removing the pages in question after she became aware that they were not appreciated by some of her colleagues, she was charged with the breach of the PUL (Swedish law on personal data), on the grounds that she had:

- "processed personal data by automatic means without giving prior notification to the Datainspektionen (Paragraph 36 of the PUL);
- processed sensitive personal data (injured foot and half-time on medical grounds) without authorisation (Paragraph 13 of the PUL);
- transferred processed personal data to a third country without authorisation (Paragraph 33 of the PUL)." [17]

Mrs. Lindqvist was eventually ordered to pay a fine, a sentence against which she subsequently appealed. As the Göta Court of Appeal had its doubts as to the interpretation of the Community law applicable in this area, it decided to stay proceedings and refer a number of questions to the ECJ for a preliminary ruling. Among other things, the ECJ ruled that:

> "The act of referring, on an internet page, to various persons and identifying them by name or by other means, for instance by giving their telephone number or information regarding their working conditions and hobbies, constitutes the processing of personal data wholly or partly by automatic means within the meaning of Article 3(1) of Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data." [17]

With this ruling, the ECJ established that the act of posting information about other individuals on your own, private website and identifying them by name or by other means, constitutes the processing of personal data, and is therefore covered by the European Data Protection Directive.

Article 3 (2) of the EU Directive states that the Directive shall not apply to the processing of personal data "by a natural person in the course of a purely personal or household activity." [16]). The Norwegian Data Inspectorate had up until the ruling by the ECJ taken this directive to include all personal websites. However, as a consequence of this new interpretation, several websites which up until then had been considered as strictly private by the Norwegian Data Inspectorate, and thus not covered by the Directive, was instead to be considered as being covered by the Directive, and therefore also by the Norwegian Personal Data Act.

## 2.2 Privacy in Computing Applications

An effective way of improving the privacy aspect of a system or application is to incorporate privacy into the design itself. In this context we generally talk about two concepts: Privacy Enhancing Technologies and Privacy by Design. Both of these concepts will be explained in the following sections.

### 2.2.1 Privacy-Enhancing Technologies

Privacy-Enhancing Technologies (PETs) usually refer to various technologies that are incorporated into a design with the express purpose of improving the privacy aspect of it. The term can be traced back to 1995, when it appeared as the title of a report commissioned by the Information and Privacy Commissioner of Ontario, Canada and the Dutch Data Protection Authority [18, 19]. The report introduced the concept of an "identity protector", and described how privacy technologies could be used to separate one's true identity from the details of one's transactions and communications, thereby leading to far fewer collections of identifiable information and thus enhancing the protection of privacy. Despite the term being coined in 1995, PET as a research topic was actually started as early as 1981 by David Chaum, when he described a method of achieving anonymous and unobservable communications over a network [20]. Over the last decades, PETs have been an active research topic in computer science. For example, in [21], Deswarte et al presents an overview of several current (2006) and future solutions for privacy protection on the Internet.

The European Commission, in its "Communication From the Commission to the European Parliament and the Council on Promoting Data Protection by Privacy Enhancing Technologies (PETs)" describes a PET as:

> "a coherent system of ICT measures that protects privacy by eliminating or reducing personal data or by preventing unnecessary and/or undesired processing of personal data, all without losing the functionality of the information system." [22]

Another much cited definition, according to Thomas Olsen [4], comes from Herbert Burkert:

> "The term privacy-enhancing technologies (PETs) refers to technical and organizational concepts that aim at protecting personal identity." [23]

Olsen explains that the core concept of PETs has always been to limit the ability to identify the data subject (cf. Burkert's definition above). He suggests that this focus on identity in relation to PETs can be explained by the fact that privacy interests does not apply unless the person related to the information can be identified. This is reflected in the Norwegian Personal Data Act, as well as the European Data Protection Directive, which only applies in cases where information can be linked to an identified or identifiable individual.

Although the original meaning of the term PETs was limited to personal identity protection, several later definitions of the term seems to consider PETs to include any technology that helps to protect or enhance an individual's privacy. There is some debate, though, on the exact scope of the term. And while there isn't a widely accepted single definition of the term, most definitions (e.g. [24, 25, 22]) seem to encapsulate the same basic principle; technology for protecting or enhancing an individual's privacy by minimizing collection and processing of personal data, as well as providing a degree of anonymity.

Both the European Union and the Norwegian government has expressed their support for PETs, viewing it as an important strategy for the protection of privacy. In "Stortingsmelding nr 17" (Article 8.3.5) [26], the Norwegian government stated that it wanted to strengthen its focus on the use of PETs. There has also been a clear political will from the EU to promote the use of PETs. In a report from 2003 [27] on the implementation of the Data Protection Directive (95/46/EC), the European Commission considered that: "the use of appropriate technological measures is an essential complement to legal means and should be an integral part in any effort to achieve a sufficient level of privacy protection." In a "Communication From the Commission to the European Parliament and the Council on Promoting Data Protection by Privacy Enhancing Technologies (PETs)" from 2007 [22], the European Commission stated its support to PETs, saying that it "considers that wider use of PETs would improve the protection of privacy as well as help fulfil data protection rules."

However, despite the support given to the concept of PETs as an important strategy for the protection of privacy, and despite being an active research topic in computer science, it seems PETs haven't really caught on and become the solution to the privacy challenges posed by the ever-growing Information and Communication Technologies that many were perhaps hoping for.

Ann Cavoukian, Ontario's Information and Privacy Commissioner, argues that these days a more substantial approach is required, by extending the use of PETs to what she calls PETs *Plus* [28]. According to her, the "plus" is meant to signify a full functionality approach to privacy, instead of the zero-sum paradigm that has prevailed over the years, where trade-offs have been made between privacy and other critical requirements such as security and usability.

A concept that has gained increasing support in later years, and is seen by many as an essential step towards better privacy protection, is what is known as *Privacy by Design*. This concept will be explained in the following section.

### 2.2.2 Privacy by Design

Privacy by Design (PbD) is an approach whereby privacy and data protection compliance is embedded proactively into system design, rather than being bolted on afterwards.

PbD is often associated with Privacy Enhancing Technologies (PETs). But while PbD is the philosophy of embedding privacy proactively into the technology during design, PETs can be seen as more of an afterthought in product design and usually takes the form of a product add-on that can be implemented into already existing technologies and systems.

According to Ann Cavoukian, the person credited for initially developing the concept, the objectives of Privacy by Design is primarily to ensure privacy and to gain personal control over one's information. She suggests the following 7 principles for accomplishing these objectives [28]:

1. Proactive, not Reactive; Preventative, not Remedial

   The PbD approach is characterized by seeking to prevent privacy infractions from occurring rather than wait for them to occur.

2. Privacy as the Default

   No action should be required on the part of the individual to protect their privacy; rather, it is built into the system by default, ensuring that personal data is automatically protected.

3. Privacy Embedded into Design

   According to the PbD approach, privacy should be an essential component of the core functionality of a system.

4. Full Functionality - Positive-Sum, not Zero-Sum

   PbD seeks to accommodate all legitimate interests, avoiding trade-offs such as privacy vs. security or privacy vs. functionality.

5. End-to-End Life cycle Protection

   By having privacy embedded into the system from the start, it can extend throughout the lifecycle of any data involved.

6. Visibility and Transparency

   By making the component parts and operations of a technology or business practice transparent and visible to users and providers alike, PbD seeks to assure all stakeholders that the stated promises and objectives are kept and can be subject to independent verification.

7. Respect for User Privacy

   PbD requires operators and system architects to keep the interests and needs of the individual user uppermost.

To the extent that they have been relevant, these principles have been sought followed in the development done as part of this thesis.

## 2.3 Web Search Engines

This section will give a brief explanation of how web search engines work. Understanding how they work and operate will help webmasters and publishers choose the right methods and tools for controlling personal data online.

Web search engines are an ingrained part of the Internet. They provide an easy and effective way of locating information on the World Wide Web, provided that the information is available to the search engines.

The term "search engine" can be used to refer to different types of searching technologies. In this context, we are talking about web search engines, i.e. software code designed to search for information on the World Wide Web. While there is no single, agreed upon definition, we will, for our purposes, use the definition put forth by Halavais (cited in the Stanford Encyclopedia of Philosophy [29]), who defines a (Web) search engine as "an information retrieval system that allows for keyword searches of distributed digital text."

In general, a search engine operates in the following order:

1. Web crawling
2. Indexing
3. Search and retrieval

Web search engines work by storing information about web pages retrieved by a robot (also known as a spider, bot, web crawler, and a number of other names). A robot is an automated computer program that browses or "crawls" the World Wide Web in a methodical fashion. While these programs can be used for a wide range of tasks, in this context the robot's function is to visit

web pages and store the information it finds for later processing. The robot navigates a website by following links on each web page (if there are any), or by using the Sitemap of the website (if one is provided). Likewise, new websites are discovered by the robot by following the links it finds on each website visited. Without further assistance, this way of navigating the web limits how much of the web a robot can discover, in that if a webpage is not linked to from pages on the "known" web, the robot cannot discover it on its own. However, most search engines provides a way around this problem by offering a Sitemap service, where a site owner can submit a Sitemap of their website directly to the search engine. Some search engines also allow a site owner to submit a link to his or her website in order to request the robot to visit and index one or more pages on their site. This can be used for webpages that hasn't been visited by a robot yet, either because it is very new or because the robot missed it for some reason, or for pages that are already indexed by the search engine, but has received an important update which hasn't yet been reflected in the search engine results pages. In theory, this could also help reduce the time it takes for a robot to discover a webpage and index it.

Robot typically identify themselves to a web server using the User-agent field of a HTTP request. They often also include a URL and/or an e-mail address, allowing the website administrator to contact the operator of the robot. The User-agent string format is specified by section 14.43 of the RFC 2616 (HTTP/1.1) [30].

The content collected by the robot is handed off to the indexer, which in turn builds a searchable list of terms found within the collected content.

When a user submit a search query to the search engine, the query processor compares the search query to the index and retrieves the pages considered most relevant. Search engines typically use an algorithm for assigning importance to indexed items. One such well-known algorithm is PageRank (Google). Such algorithms are usually closely guarded secrets, as the relevance and quality of the algorithm is often the deciding factor when people choose which search engine to use, but also to prevent people from specifically creating pages to get better ranks. Many search engines also offer users the ability to cut off or narrow the search. The resulting set, which generally contains links to all the resources that met the search specifications, is then presented in order of presumed relevance to the user. This set of results is often referred to as the search engine results pages (SERP).

In addition to a link pointing to a specific resource, some search engines store a copy of some or all of the source page as well (referred to as caching). This provides users with the choice of either visiting the original page, or view the cached copy. The latter option offers the advantage of accessing the content of the desired web page even if the original page is temporarily down or inaccessible, although the risk is that the cached page is an outdated version. Even so, this isn't necessarily a bad thing, as it gives users the

opportunity to view the content that was originally indexed, in the event that the content has since been changed. Other advantages of caching can be highlighted keywords and faster download times.

The three major search providers today, outside China, are Google (Google), Microsoft (Bing) and Yahoo (Yahoo! Search). According to Netmarketshare [31], as of April 2012, Google is the dominant internet search provider with a worldwide market share of 79.7%. Yahoo has a market share of 6.5%, and Bing has a market share of 4.5%. The Chinese search engine Baidu actually has a market share of 7.2%, which is more than Yahoo and Bing, but it is also almost exclusively used in China.

## 2.4 Summary

In this chapter we have explained some of the relevant concepts related to privacy and data protection. We have also provided an overview of the most important laws and regulations governing privacy and data protection, focusing on Norway and the European Union. We have explained two important concepts related to privacy in computing applications: *Privacy Enhancing Technologies* and *Privacy by Design*. Finally, we have attempted to clarify how search engines work and operate. In the next chapter, we move on to describe the research methods utilized in this thesis.

# Chapter 3

# Research Methods

In this chapter we describe the research methods utilized in this thesis with respect to achieving the research objectives presented in chapter 1.

In Section 3.1 we describe the design process of our solution. Then in Section 3.2 we present the method used to evaluate the Robots Exclusion Protocol. Finally, in Section 3.3, we describe the methods utilized in order to evaluate our solution.

## 3.1 Designing a Solution

The main goal of this thesis was to create a tool for increasing the control users have over personal data that is published online. The first step was to clarify the concepts of privacy and data protection. Suggested definitions for these terms, along with a brief overview of relevant laws and regulations governing privacy and data protection, are presented in Chapter 2. The next step was to investigate what methods are available for controlling how search engines crawl and index content published online. The result of this investigation is described in Chapter 4.

We then looked at two of the most popular publishing platforms today: Drupal[1] and WordPress[2]. In the end we decided to focus on the WordPress software, and develop a privacy-enhancing plugin that could be installed and used on any WordPress site. The process of designing a solution for the WordPress platform involved taking a closer look at how WordPress supports users in the task of controlling personal data, and how the software supports the development of additional functionality. This investigation is described in Chapter 5. Based on this investigation, we decided on a set of requirements that our solution should meet in order to address the privacy-related issues we discovered, before starting the process of designing and implementing a solution. Chapter 6 describes the finished product. After a working solution

[1] `http://drupal.org/`
[2] `http://wordpress.org/`

had been designed and implemented, the solution was evaluated in terms of functionality and usability. This evaluation is described in Chapter 7.

## 3.2 Evaluating the Robots Exclusion Protocol

In Chapter 4 we take a closer look at what is known as the Robots Exclusion Protocol. As part of this investigation, we wanted to conduct an empirical study on the Robots Exclusion Protocol, using a quantitative approach. As explained in *Experimentation in Software Engineering: An Introduction*, the aim of quantitative research is

> "to identify a cause-effect relationship. The quantitative research is often conducted through setting up controlled experiments or collecting data through case studies. Quantitative investigations are appropriate when testing the effect of some manipulation or activity." [32, p. 7]

The empirical strategy we chose for this study was to conduct an experiment, where the aim was to evaluate how effective this method actually is when it comes to helping website owners control how search engine robots crawl and index their content.

The experiment was designed in accordance with the process described in [32]. There, the different steps (or activities) involved in carrying out an experiment are: definition, planning, operation, analysis and interpretation, and presentation.

The first step involved defining the experiment in terms of problem, objective and goals. The next step was the planning phase. This involved formulating hypotheses for the experiment, and designing the tests that would be conducted. After the experiment had been designed and planned, we entered the operation phase, where the experiment was carried out. This involved creating two websites, each consisting of various *directives*, and then wait for search engines to come visit them. Afterwards, the data from the experiment, consisting of the search results generated by submitting search queries to the search engines, was collected and analyzed. The findings from this experiment are presented in Chapter 4.

## 3.3 Evaluating the Solution

In order to evaluate the solution presented in Chapter 6, we used two different approaches - software testing and usability testing. The software testing was conducted in order to verify that the solution worked as expected. The usability test was conducted in order to find out whether or not the solution was usable by the intended user group.

### 3.3.1 Software Testing

In order to evaluate the solution in terms of functionality, we conducted a software test. This involved setting up an experimental WordPress website, install the application, publish some content, and then let the search engine Google crawl the website. By analyzing the resulting listings in Google generated from search queries, we would be able to see whether or not the solution functioned according to the stated objective, which is to be able to separate personal names from the contextual data attached to them when the content is indexed by a search engine. The test also involved looking at the content from the viewpoint of a search engine robot, by changing the browser user-agent into that of Googlebot[3]. The evaluation is described in more detail in Chapter 7.

### 3.3.2 Usability Testing

In order to evaluate our solution in terms of usability, we conducted a *usability test*. According to Sharp et al, usability testing is:

> "an approach that emphasizes the property of being usable, i.e. it is the product that is being tested rather than the user [...] The goal is to test whether the product being developed is usable by the intended user population to achieve the tasks for which it was designed." [33, p. 646]

Data gathering techniques used for this study were observation and semi-structured interviews. According to [33, p. 321], "observation conducted later in development, e.g. in evaluation, may be used to investigate how well the developing prototype support [the users'] tasks and goals". The observation conducted in this study took place in a controlled environment, and the participants were observed directly as they performed a given set of tasks. One of the problems with observation is that the observer does not know what the user is thinking while performing the tasks, so in order to better understand what goes in the participant's head while performing the tasks, we used the *think-aloud* technique. In the article *Thinking-aloud in user interface design: a method promoting cognitive ergonomics*, Jørgensen describes the method as:

> "the thinking-aloud method consists in having a user working with a computer system (prototype, paper mock-up or documentation) while 'thinking-aloud' i.e., spontaneously (or prompted) verbalizing ideas, facts, plans, beliefs, expectations, doubt, anxiety, etc. that comes to mind during the work. Typically a scenario is developed for the tests, i.e., an artificial work context

[3]The search engine robot operated by Google

> with specific tasks that can be accomplished by means of the system." [34, p. 502]

For our test, we wanted to see how users accomplished a series of typical tasks when using our solution, such as tagging a personal name and change some of the plugin settings. Before the testing began, the participant was told to "think aloud" while performing the tasks. In the article *Thinking Aloud: Reconciling Theory and Practice* [35], several different situations are mentioned as possible to find oneself in during such a test, and suggests how to accommodate the different situations. Some of these situations did in fact occur during testing. For example, several times the participants performed a task/did something on the screen, and then looked at the researcher for some sort of confirmation that they had done it right, or asked the researcher outright whether or not they had done it right. These verbal or non-verbal questions were usually answered with uncommitted acknowledgment tokens, such as "mm hm" or "uh-huh" followed by an interrogative intonation. According to [35], these are the most appropriate acknowledgment tokens during usability tests, as they "rarely occur at changes in speakership, and when given an interrogative intonation, they may seamlessly invite the participant to "please continue" - all without introducing any task-relevant content, diverting attention to anything in particular, or requiring further elaboration on the participant's part". Sometimes the participant would stop thinking aloud. In these cases, simple remainders were given, usually on the form of "what task are you working one now?", "what are you thinking now?" or, as suggested in [35], by extending an acknowledgment token such as "mm hm?" even though there is nothing to acknowledge. Three times during the tests, a situation arose where a participant mistakenly thought that a task was completed, and moved on to the next task. In all three cases, the researcher opted not to interfere, and instead let them continue on with the tasks. In one of these cases, however, this meant that the following tasks proved impossible to complete. In this case, the researcher waited to see if the participant would realise that they had to have missed something in a previous task and backtrack. After a certain amount of time had passed, the researcher subtly hinted that maybe they had missed something in a previous task. The participant then re-read the previous tasks, realised where they had gone wrong, and was able to correct their mistake.

The usability testing was documented by means of audio recording and screen capture. The evaluation is described in more detail in Chapter 7.

# Chapter 4

# Robot Directives

There are some cases in which webmasters wish to communicate some information to search engines, for example that they don't want certain content to be indexed. For this purpose they can use what is commonly called the *Robots Exclusion Protocol* (REP), also known as the *Robot Exclusion Standard* (RES) or the *robots.txt protocol*. This is a collection of guidelines that regulate the behaviour of web robots and search engine indexing. However, there is no official standards body or RFC (Request for Comments) for the REP. It merely represents a consensus that was reached in June 1994 by the members of the robots mailing list (robots-request@nexor.co.uk) [36]. This can lead to potential problems when relying on the REP as a method for constraining search engine robots. For example, different search engines may interpret the directives differently, and some directives may not be supported at all. Also note that the REP is not in any way enforceable, and it is up the search engine robots themselves whether or not they want to adhere to the directives, although the well-behaved ones usually do. This means that other methods will have to be employed if one wishes to constrain robots that are not so well-behaved, for example by using a *.htaccess* file to deny access to any robots which ignores or disobeys the REP directives.

While the original REP from 1994 only defined crawler directives for the *robots.txt* file (which then only included the *Disallow* directive) [36], the protocol has evolved over the years, as the major search engines have come together to agree on new directives, or introduced their own directives which are only supported by that search engine. These extensions includes mechanisms for controlling what content *should* be included (which was difficult to accomplish with only the *Disallow* directive), as well as how the content is displayed in the search engines' result page and how frequently the content is crawled.

Crawling and indexing are two different processes, and it is important to understand the difference between them in order to properly control how your content is displayed in search engines. This distinction is further reflected

in the fact that there are basically two types of directives, depending on whether you want to communicate to the search engines how their robots should crawl your site, or how they should index it. These two types of directives are further explained in the following sections.

## 4.1 Controlling Crawling

The crawling directives are placed in a file called *robots.txt*, which must be placed in the root directory (i.e. the highest-level directory) of a website. If you have multiple subdomains, the robot must be able to fetch the robots.txt file at the root of each one. If a robots.txt is missing for a subdomain, the robot will not attempt to fall back to other robots.txt, meaning that the robot will consider itself allowed anywhere on the subdomain.

The way it works is, if a robots wants to visit a website URL, for example http://www.examplewebsite.com/index.html, it will first check to see if the website contains a robots.txt file, which in this case will be located at http://www.examplewebsite.com/robots.txt (provided that the robot is well-behaved and actually adheres to the REP).

### 4.1.1 Content of the robots.txt file

The format and semantics described here is based on information given in the original *A Standard for Robot Exclusion* document from 1994 [36], information provided by the robotstxt.org website [37], and various search engine support pages.

The *robots.txt* file contains one or more records or sections, separated by one or more blank lines. Each record consists of one or more *User-agent* lines, followed by one or more directive lines. Each line is on the following form:

```
<field>:<optionalspace><value><optionalspace>
```

The *value* of the User-agent field is the name of the robot the record is describing the access policy for, or '*' to denote all robots. According to *A Standard for Robot Exclusion* document from 1994 [36], it is not allowed to have multiple records with '*' as the value for the User-agent field. However, you can have one record which describes the access policy for all robots, and then one or more records which addresses specific robots. The *field* element denotes the directive, while the *value* element is a partial URL, which can be a full path or a partial path. Also note that at least one *Disallow* field needs to be present in a record. The two lines below shows an example:

```
User-agent: *
Disallow: /tmp/
```

Table 4.1: Crawler directives

| **Directive** | **Function** | **Supported by** |
|---|---|---|
| Disallow | The most common directive, and the one most search engines seem to support. Used to suggest to web crawlers which pages or directories they should avoid crawling. | Google, Bing |
| Allow | Some major search engines support an *Allow* directive, used to refine previous *disallow* statements by allowing certain files to be crawled in an otherwise disallowed directory. | Google, Bing |
| Sitemap | If you have created an XML-based sitemap for your site, you can add a reference to its location with this directive. | Google, Bing |

The first line indicates that the following directive addresses all robots. The second line tells the robots that they are not allowed to crawl the */tmp/* directory.

Some search engines are more strict than others when it comes to interpreting the content of a robots.txt file. For example, simple errors or typos (e.g. "useragent" instead of "user-agent") may result in some search engine robots disregarding that record completely, while others may try to interpret the record anyway. Likewise, if there should happen to be some logical confusion and both *Allow* and *Disallow* directives apply to the same URL, search engines may decide differently on which directive should take precedent. In addition, to ensure compatibility with the largest number of search engines, Bing recommends listing all *Allow* directives before the generic *Disallow* directives for the same directory [38].

Table 4.1 lists the crawler directives which are officially supported by the major search engines (i.e. Google [39] and Bing [40]). Some search engines also supports a limited form of wild cards for path values. These are "*", which represents 0 or more valid characters, and "$", which designates the end of the URL.

## 4.2 Controlling Indexing and Serving

While the *robots.txt* can be used to tell a web robot not to crawl a specific file of directory, there are times when we want the URLs to show up in the

search engine result page (SERP), but we also want to prevent a cached version of our webpage or website from appearing in the SERP. This can be accomplished using so called indexer directives.

Indexer directives are used to set directives on a per page or per element basis, instructing search engines how to index the content. In order for a search engine to be able to comply with an indexer directive, they must be allowed to crawl the resource that provides the indexer directive. This means that the crawler cannot be disallowed from crawling the resource using the robots.txt file. Otherwise the crawler won't be able to find and read the directives specified for that resource.

These directives are defined either within the HTML code of each web page, by adding the **robots meta tag** to the `<head></head>` section, or as attributes in the `<a>` tag; or by adding the **X-Robots-Tag** to the HTTP header response for a given URL. The following is a sample meta tag that addresses all robots, telling them not to index the content of a page:

```
<meta name="robots" content="noindex">
```

In this example, the *name* attribute indicates that the directive should be read by a robot when it accesses the page, while the directives specified in the *content* attribute tells the robot what (not) to do with the content. In addition to being described on the support pages of search engines like Google and Bing (e.g., [41, 42, 38]), the robots meta tag is also described in the HTML 4.01 specification, Appendix B.4.1 [43].

Table 4.2 lists the values of the robots meta tag currently supported by the two major search engines today: Google [42] and Bing [38].

Not all content types allow you to add meta tags (e.g. PDF files). Instead you can modify the HTTP header to include the X-Robots-Tag, allowing you to define REP directives for those content types as well. According to Google [42] and Bing [38], the directives are the same as for meta tags, meaning that any directive that can be used in a robots meta tag can also be specified as an X-Robots-Tag.

## 4.3 Related Work

Despite the importance of the Robots Exclusion Protocol (REP), there seems to be relatively few studies investigating its usage in detail. The first reasonably large-scale study of its usage was done as late as 2007 [46, 47], and involved crawling 7593 unique websites, covering the domains of education, government, news, and business. The study revealed many incorrect uses of the Robots Exclusion Protocol, in addition to several robots.txt files containing ambiguous and conflicting rules. The study concludes that a better-specified, official standard is needed.

In an article from 2008 [48], Kolay et al. performed a similar survey from the one in 2007, only on a much larger scale, in an attempt to correct what

Table 4.2: Indexer directives

| **Directive/Value** | **Function** | **Supported by** |
|---|---|---|
| noindex | Prevents the page from being indexed. | Google, Bing |
| nofollow | Prevents the robot from following the links on the page, but the page can still be indexed. | Google, Bing |
| none | Equivalent to noindex, nofollow. | Google |
| nosnippet | Instructs the robot to not display a snippet in the SERP. Snippets are the text descriptions that usually accompany the link to the page in the SERP. | Google, Bing |
| noarchive | Instructs the robot to not show a "cached" link for that page in the SERP. | Google, Bing |
| nocache | Same as noarchive. | Bing |
| noodp | Instructs the search engine to not use metadata from the Open Directory Project [44] to display a snippet or a title for that page in the SERP. | Google, Bing |
| notranslate | Do not offer translation for that page in the SERP. | Google |
| noimageindex | Do not index images appearing on that page. | Google |
| unavailable_after: [RFC-850 date/time] | Do not show this page in search results after the specified date/time. The date/time must be specified in the RFC 850 format [45]. | Google |

they saw as the shortcomings of the previous survey and investigate if there really was any significant bias towards specific search engines among a set of robots.txt files, as the results from the previous study had indicated [47]. The study showed that the bias towards search engines was not as serious as reported by the 2007 study.

While the number of studies investigating the usage of the protocol are relatively few, there seems to be no studies conducted on how search engines actually relate to directives given to them. The following section describes an experiment conducted as part of this thesis, where the aim was to get empirical data on exactly how search engines adhere to the REP.

## 4.4 Testing the Robots Exclusion Protocol

The Robots Exclusion Protocol (REP) is one of the few methods available to webmasters for communicating with search engine robots. For this reason it is important to have a clear idea of not just how the REP is supposed to work, but also how different search engines actually adhere to it. While the support pages of Google and Bing provide some documentation of which directives they support and how they interpret them, we decided that we needed some empirical data.

After reading the search engines' support pages explaining the REP and how they relate to it, a couple of interesting points presented themselves. For instance, after specifying that a webpage should not be crawled using the *Disallow* directive, one would perhaps not expect that page to appear in the search engine results page (SERP). However, according to Google and Bing, pages that are disallowed from being crawled may still be indexed by search engines. As Bing mentions on their Webmaster Center blog:

> "For URLs disallowed in robots.txt, search engines do not attempt to download them. Without the ability to fetch URLs and their content associated, no content is captured and no links are followed. However, search engines are still aware of the links and they may display link-only information with search results caption text generated from anchor text or ODP information." [38]

So in theory, search engines may decide to include a link to your pages in their SERP without having crawled them, as long as those pages are linked to from somewhere else. Keep in mind that search engines like Google and Bing still adheres to your robots.txt instructions by not actually indexing any of the content on the disallowed pages. While Bing does not specify under what circumstances un-crawled pages may or may not appear in their SERP, Google are somewhat more specific, saying that:

> "Pages may be indexed despite never having been crawled: the two processes are independent of each other. If enough information is available about a page, and the page is deemed relevant to users, search engine algorithms may decide to include it in the search results despite never having had access to the content directly. That said, there are simple mechanisms such as robots meta tags to make sure that pages are not indexed." [41]

So, according to Google, the deciding factors for whether or not an uncrawled page may or may not appear in their SERP are **(1)** the information available about it and **(2)** to what degree the page is deemed relevant to users. Google does not specify what would be considered "enough" information about a page for it to appear in their SERP. Nor do they give any examples of what would make a page relevant enough to users that the search engine algorithm would decide to include the page in their SERP. It is also unclear what Google means when they talk about "information" being available about a page, but it probably means information generated from anchor text, information taken from the Open Directory Project (ODP), or information found on other pages that are linking to it.

The theory that pages could be indexed without having been crawled was one of the things we wanted to test, as well as what it would take for a disallowed page to appear in the SERP. For instance, would a single link to a page that is blocked from crawling be enough for that page to show up in the SERP as a URL-only listing? To this end, an experiment was conducted by setting up a couple of websites with various directives, wait for the search engine robots to visit them, and then investigate how the search engines related to any directives they encountered.

By accumulating empirical data regarding search engines' adherence to the REP, this experiment may hopefully provide webmasters and publishers with a better understanding of how crawlers and search engines relate to directives given to them through the use of the REP, as well as some useful pointers about how they can keep certain content from being indexed.

The next section lists the questions that we wanted our experiment to answer. Section 4.4.2 on the following page describes how the experiment was set up and carried out, while the results are presented in section 4.4.3 on page 33. The results from the experiment are then discussed in Section 4.5.

### 4.4.1 Aim of the Study

The experiment conducted was aimed at answering the following question:

- P: How reliable is the REP as a method for keeping content out of search engines?

This was further divided into the following two sub-questions:

- $P_1$: Can we trust that search engines adhere to the directives they are given?
- $P_2$: When using the REP, what is the most effective way of keeping pages from being indexed by search engines: the *Disallow* directive in a robots.txt file, the *noindex* meta tag, or a combination of both?

By answering question $P_1$ and $P_2$, we would hopefully also be able to answer the main question P.

### 4.4.2 Method

As mentioned in Chapter 3, the empirical strategy chosen for this study was to conduct an experiment. It was designed in accordance with the process described in [32]. There, the different steps (or activities) involved in carrying out an experiment are: definition, planning, operation, analysis and interpretation, and presentation (Note that not all of these steps are described here).

In order to answer the first sub-question ($P_1$) stated in Section 4.4.1, the following null hypothesis and alternative hypothesis was formulated for this experiment:

- $H_0$: Search engine robots will *not* follow the directives they find, *even though* the search engines in question officially support them, and the robots can actually read the directives.
- $H_1$: Search engine robots will follow the directives they find, given that the search engines in question officially support them, and that the robots can actually read the directives.

The formulation of this alternative hypothesis is based on the fact that both Bing and Google officially state that they support directives provided in a robots.txt file, within meta tag or as attributes in the <a> tag, or within the HTTP Header from the web server (e.g., [38, 41]).

To answer the second sub-question ($P_2$) stated in Section 4.4.1, the following null hypothesis and alternative hypothesis was formulated:

- $H_0$: Given that the web page is crawled by the search engine robots, the page will *not* appear in the SERP as a URL-only listing if it is disallowed in a robots.txt file.
- $H_1$: Given that the web page is crawled by the search engine robots, the page will appear in the SERP as a URL-only listing *despite* being disallowed in a robots.txt file.

The formulation of this alternative hypothesis is based on the information found on the support pages of Google and Bing, where they state that "without the ability to fetch URLs and their content associated, no content is

captured and no links are followed. However, search engines are still aware of the links and they may display link-only information [...]" [38] and "Pages may be indexed despite never having been crawled: the two processes are independent of each other" [41].

According to [33, 32], a hypothesis typically involves examining a relationship between two variables. Variables can be independent or dependent. According to [32], "the independent variables are those variables that we can control and change in the experiment", while "the effect of the treatments is measured in the dependent variable or variables." In this experiment, we wanted to test how the search engines would react to the various directives they encountered on our websites. So the independent variables here would be indexing directives and crawling directives, while the dependent variable would be the search engine robots. Also mentioned in [32] is that the dependent variable "is mostly not directly measurable and we have to measure it via an indirect measure instead." In this case, the indirect measure would be the SERP of the various search engines.

### Procedure

The experiment was split into four test cases: **(1)** Check whether the order the crawler directives are written in makes any difference in how search engine robots interprets the directives. **(2)** Test which crawler directive takes precedent if there is some logical confusion and both Allow and Disallow directives apply to the same URL within a robots.txt file. **(3)** Check what it will take for a web page to be indexed even if it has been disallowed in a robots.txt. **(4)** Confirm that search engine robots adhere to directives.

For the purpose of this experiment, two websites were created (*Website Alfa* and *Website Beta*), each consisting of a set of webpages with different directives (or combinations of such). Website Alfa would be used for the first and second test case described above, while Website Beta would be used for the third and fourth test case.

All the webpages in this test were similar in content and structure, although no two pages were identical. The reason for this is that Google (and possibly other search engines as well) filters out duplicate content in their listings, or, if they perceive that duplicate content may be shown with intent to manipulate and deceive their users, make "appropriate adjustments in the indexing and ranking of the sites involved. As a result, the ranking of the site may suffer, or the site might be removed entirely from the Google index, in which case it will no longer appear in search results" [49].

Each page consisted of a `<head></head>` section (containing a title, and sometimes one or more robots meta tags), a heading, and one or two short paragraphs taken from the novel *Alice's Adventures in Wonderland* by Lewis Carroll. In addition, some of the pages contained a list of links to internal or external pages (all of which were part of the same experiment). Each

website also had a Sitemap located in the root directory, created with a free online Sitemap generator[1]. These were generated in the XML Sitemaps format according to Sitemap protocol 0.9[2]. All the pages tentatively passed the W3C Markup Validation Service[3] as HTML 5 (the only issue found was a lack of information regarding character encoding).

After the websites had been created, a URL to both websites was submitted to Google and Bing (along with a link to the Sitemap for each website). Both Google[4] and Bing[5] provide this option for site owners using their Webmaster Tools. This was mainly to let the search engines know that the sites existed, but the webmaster tools also provided access to reports and data regarding crawling and indexation of the websites.

Note that while this experiment was designed to involve the three major search engines *Google*, *Bing* and *Yahoo! Search*, the latter is currently being powered by Microsoft's search engine Bing [50]. Therefore, although Yahoo is not further mentioned in this chapter, the test results for Bing could also be applied to Yahoo! Search. However, while it didn't take very long for the websites to be indexed by Google, it took about 9 weeks for Bing to include any of the pages in their index. Unfortunately, the only two pages that had been indexed were the homepages of both websites, which are not actually a part of any of the tests, as they are not the subject of any directives. None of the pages in this experiment have been submitted to the search engines on an individual basis. Only the primary domain name of each site has been submitted. It is unknown whether or not submitting each individual page to Bing would have helped getting them indexed, but this was not done in order to avoid risking influencing the test results. And in any case, the presence of a sitemap should in theory accomplish the same thing. On their support pages [51], Bing suggest several reasons why a site does not show up in their index. Of those, the following three reasons seems the most likely in this case:

- Not enough links are pointing to the websites from other places on the web, indicating to Bing that these are not very popular sites.
- The websites does not meet the quality threshold required by Bing (after all, these sites were not created with content quality in mind).
- Bing is just being extremely slow at indexing.

Whatever the reason may be, this means that only Google is represented in the test results.

The following is a more detailed description of each website.

[1] `http://www.xml-sitemaps.com`
[2] As defined by `http://www.sitemaps.org/`
[3] `http://www.w3schools.com/web/web_validate.asp`
[4] `http://www.google.no/webmasters/`
[5] `http://www.bing.com/toolbox/webmaster/`

### Website Alfa

This website consisted of a set of 12 web pages, including a homepage. The following meta tags were placed in the <head></head> section of some of the web pages:

```
Page 1: <meta name='robots' content='noindex,nofollow' />
Page 2: <meta name='robots' content='noindex' />
Page 3: <meta name='robots' content='nofollow' />
Page 4: <meta name='robots' content='nosnippet' />
Page 5: <meta name='robots' content='noarchive' />
```

Page 1 and page 3 contained an outgoing link to pages that were not linked to from anywhere else (page 10 and page 11), in order to test the <nofollow> tag. Pages 1 - 5 and 8 - 11 were placed directly under the root directory of the site. Pages 6 and 7 were placed in separate sub-folders (/private1/ and /private2/). The homepage contained a list of links pointing to all the pages belonging to this site (apart from pages 10 and 11), as well as links pointing to five pages which again linked to the second website in this test (Website Beta). The anchor text for all the links pointing to internal pages on this website were so called descriptive, and consisted of the same text as the headline on the page they were leading to (i.e., the text between the <h1></h1> tag). When we speak of anchor text, we mean the clickable text in a hyperlink. Figure 4.1 shows a sitemap for the website, while Table 4.3 gives a more detailed overview of the setup, as well as the results from the tests. The robots.txt file for this website contained the following:

```
User-agent: *
Allow: /private1/page6.html
Disallow: /private1/
Disallow: /private2/
Allow: /private2/page7.html
Allow: /page8.html
Disallow: /page8.html
Disallow: /page9.html
Allow: /page9.html
```

As we can see here, robots are both allowed to and disallowed from crawling page8.html and page9.html. In this case we wanted to see which directive would take precedence. We also wanted to see if the order the directives are given in has any affect on the result, hence why the order of the directives are different for the two pages.

### Website Beta

This website consisted of 20 webpages, including a homepage, of which 19 were placed in a sub-folder which was disallowed in a robots.txt file. Page 1

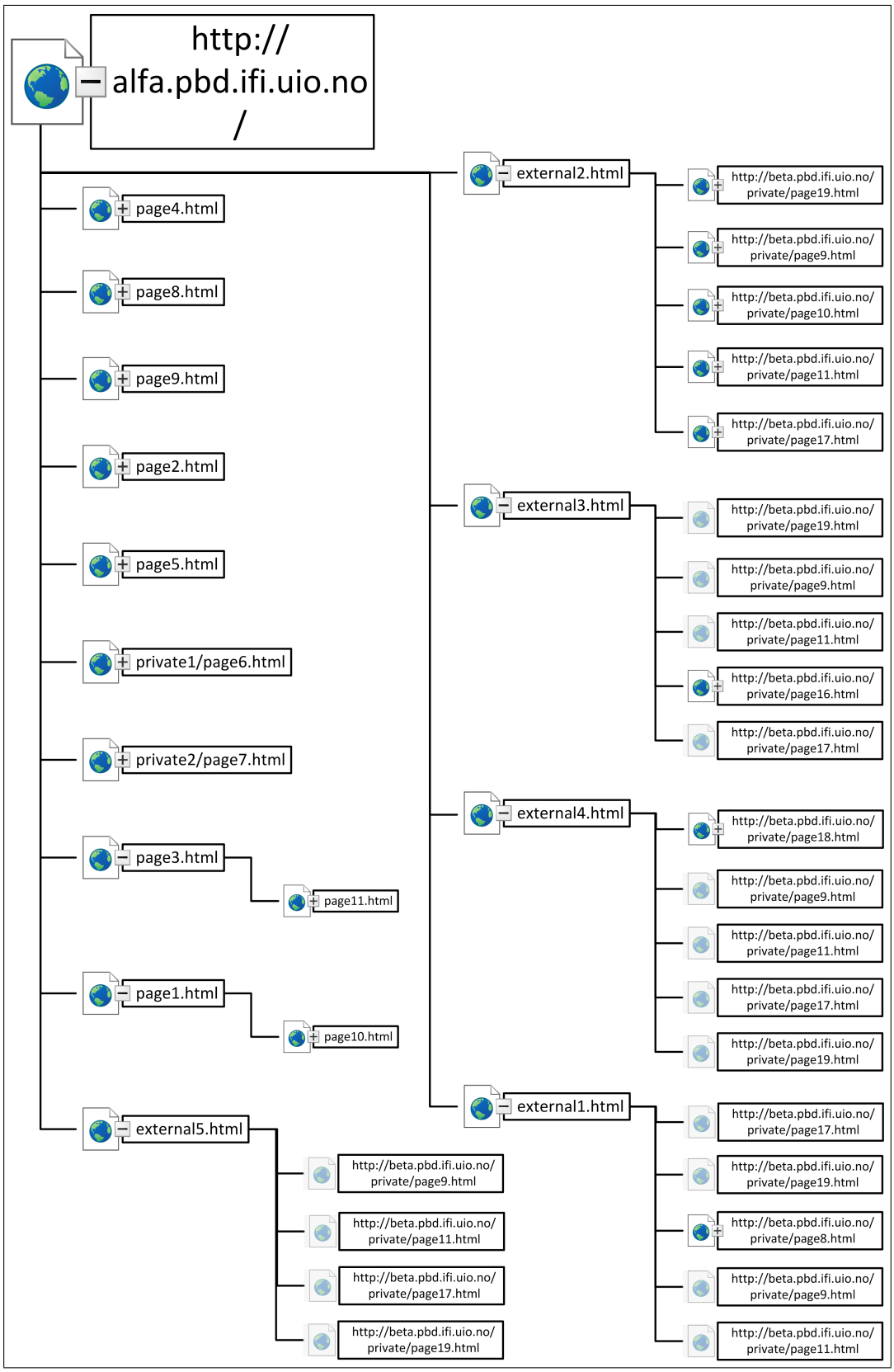


Figure 4.1: A sitemap of Website Alfa.

was not linked to from anywhere. Pages 2 and 5 were only linked to from the Sitemap. The rest of the pages, excluding the homepage, was linked to from between 1 and 5 pages (varying between internal and external pages). All of the links contained anchor text, although some of the anchor text consisted of a pure URL, while others were more descriptive and contained the same text as the headline on the page they were leading to (i.e., the text between the `<h1></h1>` tag). Pages 5, 6, 7, 10, 11, 14, 15, 18, and 19 contained the `<noindex>` robots meta tag. Figure 4.2 shows a sitemap for the website, while Table 4.4 gives a more detailed overview of the setup, as well as the results from the tests. The robots.txt file for this website contained the following:

```
User-agent: *
Disallow: /private/
```

This directive meant that all robots were disallowed from crawling the `/private/` directory.

#### Analysis

After a certain period of time had passed, a search for both websites was conducted on Google.no and Bing.com. The search query was on the form *site:<URL>*, replacing *<URL>* with the actual URL of the websites. According to the support pages of Google and Bing [52, 53], the *site* search operator can be used to help focus a search. What it does is to return all pages that belong to the specified domain. So if any of the pages appeared in the SERP, it meant that they had been indexed by the search engine. So depending on *which* pages appeared in the SERP, as well as *how* they appeared, it would tell us something about how the various directives had affected the behaviour of the search engine robot.

### 4.4.3 Findings

In this section we present the results from the experiment described in Section 4.4.2.

The results from submitting the search queries to Google[6] are listed in Table 4.3 for website Alfa and Table 4.4 for website Beta.

As we can see from the results for website Alfa, most of the directives were adhered to by Google. The only exception seemed to be the *noarchive* directive on page 5, as Google displayed a cached version of it in the SERP, despite the directive telling it not to archive it. Curiously, Google did not display a cached version of page 4, despite the absence of a *noarchive* directive. We also see that both page 6 and page 7 shows up in the SERP, indicating that the order the directives are written in makes no difference to

[6] `http://www.google.no/`

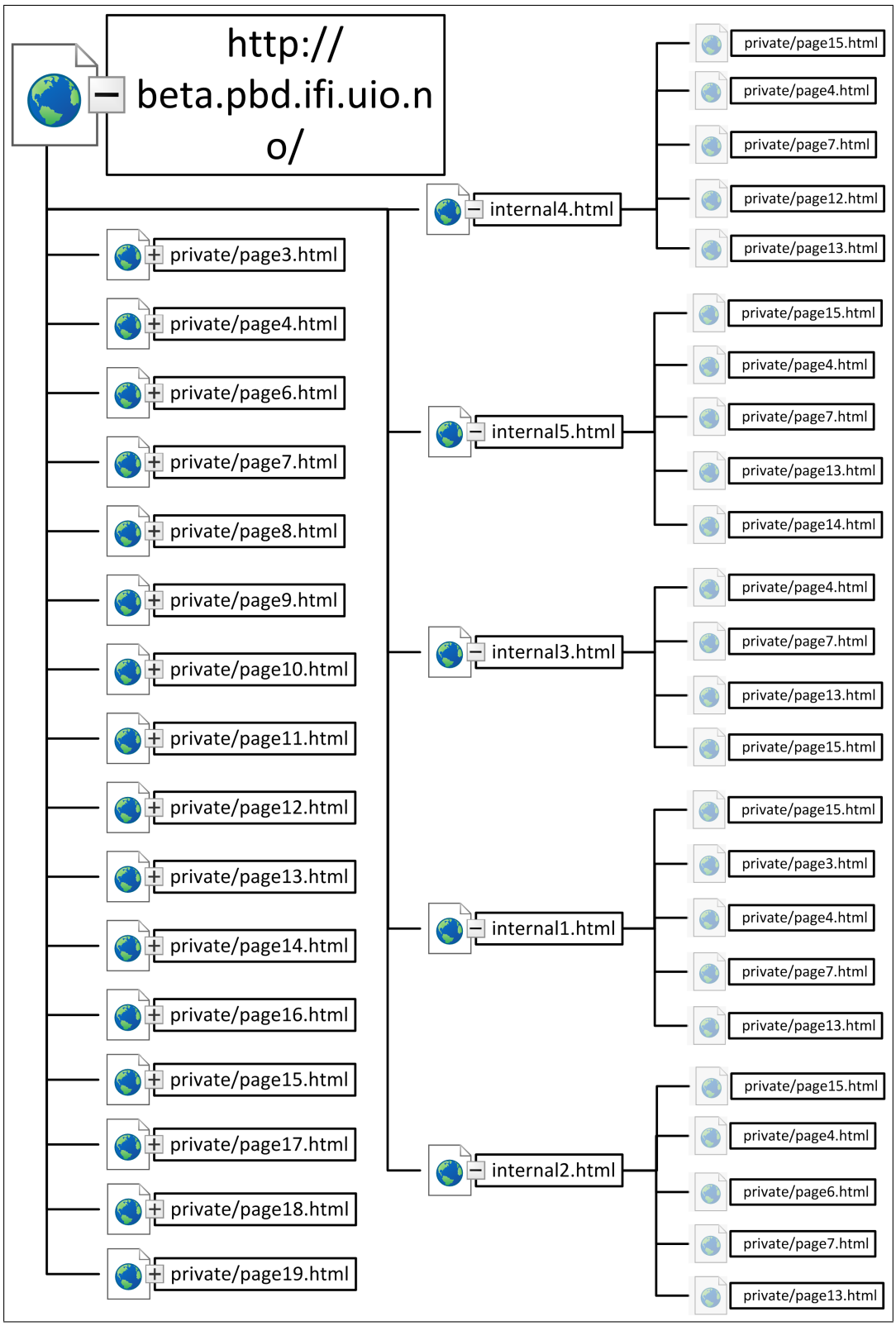


Figure 4.2: A sitemap of Website Beta.

Table 4.3: Google SERP result for website Alfa

| **Website Alfa** | | | | | |
|---|---|---|---|---|---|
| **Web page** | **robots meta tag** | **robots.txt** | **Inbound links** | **Anchor text** | **Appears in the SERP** |
| Page 1 | `noindex`, `nofollow` | - | 1 (internal) | Descriptive | No |
| Page 2 | `noindex` | - | 1 (internal) | Descriptive | No |
| Page 3 | `nofollow` | - | 1 (internal) | Descriptive | Yes |
| Page 4 | `nosnippet` | - | 1 (internal) | Descriptive | Yes |
| Page 5 | `noarchive` | - | 1 (internal) | Descriptive | Yes |
| Page 6 | - | `Allow` | 1 (internal) | Descriptive | Yes |
| Page 7 | - | `Allow` | 1 (internal) | Descriptive | Yes |
| Page 8 | - | `Allow`, `Disallow` | 1 (internal) | Descriptive | Yes |
| Page 9 | - | `Allow`, `Disallow` | 1 (internal) | Descriptive | Yes |
| Page 10 | - | - | 1 (internal) | Descriptive | No |
| Page 11 | - | - | 1 (internal) | Descriptive | No |

Table 4.4: Google SERP result for website Beta

| **Website Beta** | | | | | |
|---|---|---|---|---|---|
| **Web page** | **robots meta tag** | **robots.txt** | **Inbound links** | **Anchor text** | **Appears in the SERP** |
| Page 1 | - | `Disallow` | 0 | - | No |
| Page 2 | - | `Disallow` | 1 (Sitemap) | - | No |
| Page 3 | - | `Disallow` | 2 (internal) | Pure URL | Yes |
| Page 4 | - | `Disallow` | 6 (internal) | Pure URL | Yes |
| Page 5 | `noindex` | `Disallow` | 1 (Sitemap) | - | No |
| Page 6 | - | `Disallow` | 2 (internal) | Pure URL | Yes |
| Page 7 | `noindex` | `Disallow` | 6 (internal) | Pure URL | Yes |
| Page 8 | - | `Disallow` | 1 (external) | Pure URL | Yes |
| Page 9 | - | `Disallow` | 5 (external) | Pure URL | Yes |
| Page 10 | `noindex` | `Disallow` | 1 (external) | Pure URL | Yes |
| Page 11 | `noindex` | `Disallow` | 5 (external) | Pure URL | Yes |
| Page 12 | - | `Disallow` | 2 (internal) | Descriptive | Yes |
| Page 13 | - | `Disallow` | 6 (internal) | Descriptive | Yes |
| Page 14 | `noindex` | `Disallow` | 2 (internal) | Descriptive | Yes |
| Page 15 | `noindex` | `Disallow` | 6 (internal) | Descriptive | Yes |
| Page 16 | - | `Disallow` | 1 (external) | Descriptive | Yes |
| Page 17 | - | `Disallow` | 5 (external) | Descriptive | Yes |
| Page 18 | `noindex` | `Disallow` | 1 (external) | Descriptive | Yes |
| Page 19 | `noindex` | `Disallow` | 5 (external) | Descriptive | No |

how the crawler directives are interpreted by Google. Both page 8 and page 9 shows up in the SERP as well, indicating that for Google, Allow takes precedence over Disallow, regardless of the order they are written in. Google states on their support pages that:

> "At a group-member level, in particular for allow and disallow directives, the most specific rule based on the length of the [path] entry will trump the less specific (shorter) rule. The order of precedence for rules with wildcards is undefined." [39]

However, this author has been unable to find any place where Google specifies which directive takes precedence — Allow or Disallow — in cases where the length of the path entries are equal.

For website Beta, almost all of the pages in the private folder appeared in the Google SERP, despite being disallowed in the robots.txt. The presence of a *noindex* meta tag on some of the pages made no difference. None of the private pages that showed up in the SERP linked to a cached version, nor were they accompanied by a snippet. For the pages that were linked to with only a pure URL as anchor text, the link in the SERP did not contain the title of the page, merely the URL. All this suggests that the pages have not in fact been crawled by the Google robot.

The only pages not indexed by Google were pages 1, 2, 5, and 19. No links were pointing to page 1, and as such it is as expected that it did not show up in the SERP. Page 2 and page 5 were only linked to from the Sitemap, but this should not prevent a robot from crawling them. On the contrary, one of the main purposes of a Sitemap is to tell search engines about pages on your site that they might not otherwise discover. As stated on the Google support pages: "Creating and submitting a Sitemap helps make sure that Google knows about all the pages on your site, including URLs that may not be discoverable by Google's normal crawling process" [54]. From this, we would expect page 2 and page 5 to be treated the same way as the other disallowed pages. However, the results seems to indicate that this has not been the case. The reason why page 19 was not indexed by Google is unclear. The only thing that makes it any different from, say, page 11, is a more descriptive anchor text in the links linking to it. Apart from that, both pages were linked to from 5 external pages, both contained a noindex meta tag, and both were disallowed in a robots.txt.

It remains unclear what Google deems enough information about a web page, or on what grounds it is deemed relevant to users, in order for it to be indexed. Google does not clarify this on their support pages, and the results from the experiment does not shed any light on it. The number of links does not seem to make any difference, as even a single link pointing to a page was enough for it to be indexed despite being disallowed. Whether the anchor text in the links was descriptive or merely a pure URL seemed to make no apparent difference either.

### 4.4.4 Validity Evaluation

This experiment involved two major search engines: Google and Bing. Unfortunately, because Bing did not include us in their search results, we were only able to generate data from Google. We are therefore unable to generalize our findings to any other search engines.

An unknown factor in this test is the search algorithms used by Google. There were a couple of instances where the results were not what we expected them to be, based on the information we found on Google's support pages. Due to this unknown factor, other experiments conducted in the same way may produce different results.

## 4.5 Discussion

At the start of the experiment, we formulated both a null hypothesis and an alternative hypothesis for each of the two sub-questions. Unfortunately, we feel that we simply do not have enough data to clearly reject either hypothesis. As such, we are hesitant to draw any definite conclusions. However, we will still attempt to tentatively answer the research questions stated at the beginning of the experiment.

The aim of this experiment was to investigate the effectiveness of using the REP as a method for controlling search engine robots. As such, we set out to answer the following question:

> P: How reliable is the REP as a method for keeping content out of search engines?

This was further divided into the following two sub-questions:

> P1: Can we trust that search engines adhere to the directives they are given?
>
> P2: When using the REP, what is the most effective way of keeping pages from being indexed by search engines: the *Disallow* directive in a robots.txt file, the *noindex* meta tag, or a combination of both?

Below, we will attempt to answer both of these sub-questions in light of our findings from the experiment. By doing so, we will hopefully also be able to answer the main question.

- **P1: Can we trust that search engines adhere to the directives they are given?**

The answer to this question is dependent on whether or not the search engine officially support the REP. When conducting our experiment, we chose to concentrate on two search engines who both states their support of the REP:

Google and Bing. Unfortunately, because Bing did not include us in their search results, our answer here is limited to Google.

Based on the results from this experiment alone, one could be inclined to suggest that in most cases, Google seems to adhere to the directives given. While there is some inconsistent behavior (e.g., Google displays a cached version of a page despite a *noarchive* directive, while another page on the same website is not cached despite the absence of the aforementioned directive), the results indicate that the REP could be considered quite reliable as a method for keeping content out of Google's index, as long as the directives are used the way they are meant to be used. But again, these results are only applicable to Google. And the fact of the matter is, Google would have little to gain and much to loose from blatantly disregarding those directives. The case may be different for other search engines, especially those who do not officially support the REP, and as such may not feel obligated to adhere to some or all directives.

- **P2: When using the REP, what is the most effective way of keeping pages from being indexed by search engines: the *Disallow* directive in a robots.txt file, the *noindex* meta tag, or a combination of both?**

As Google and Bing mentions on their support pages [38, 41], in order to properly prevent indexing of a webpage, search engine robots must be allowed to crawl it. Otherwise, they won't be able to read any robots meta tags present. And as our tests have shown, disallowing a page from being crawled is no guarantee that the page won't be indexed. If we disregard the pages that were only linked to from the Sitemap, or not linked to at all, then 15 out of 16 pages ended up being indexed by Google despite the Disallow directive preventing Googlebot[7] from crawling them. In comparison, none of the pages containing a *noindex* meta tag, but which Googlebot was allowed to crawl, were indexed by Google.

While the results from the experiment are only applicable to Google, it is still a strong indication that in order to keep pages from appearing in the SERP of a search engine, it is better to use the *noindex* meta tag instead of a *Disallow* directive in a robots.txt file, provided that the search engine actually support robot meta tags. This corresponds with the fact that indexing directives are intended to be used for controlling indexing, and crawler directives are intended to be used for controlling crawling. One problem is that while it seems like a majority of search engines support crawler directives specified in a robots.txt, not all search engines support indexer directives using meta tags. In this case, one could for example use an *Allow* directive specifically for Google, and then a *Disallow* directive for all other robots.

[7] Google's web crawling bot

- **P: How reliable is the REP as a method for keeping content out of search engines?**

The answer to this question depends on two things. Firstly, not all robots are well-behaved, meaning that they deliberately disobey any directives given to them. However, the robots operated by the larger search engines are generally considered to be well-behaved. Secondly, not all search engines actually support the REP. As mentioned at the beginning of the chapter, the REP is only a de-facto standard. Different search engines may interpret the directives differently, or not support them at all. Or they may support only some parts of it (e.g., directives specified in a robots.txt file, but not directives in a robots meta tag). And because there is no official standards body or RFC for the REP, there is also no official place where information regarding usage of the REP can be found. The closest we get to an official description of it is the document describing the original consensus reached in 1994 [36], as well as a note by the W3C, describing some suggestions for making documents more accessible to search engines[8]. However, the protocol has evolved since the original consensus, with search engines introducing new directives (sometimes supported only by the search engine that introduced it) as they see fit. This makes it quite cumbersome to use directives other than the original *Disallow*, as it requires visiting each individual search engine to find out which directives they actually support. All this makes it somewhat problematic to rely purely on the REP as a method for preventing content from being crawled and indexed by search engines. However, if we focus on the more dominant search engines, like Google and Bing, the findings from our experiment suggest that the REP can be regarded as quite reliable, provided that it is used correctly. Other methods can then be used in combination with the REP in order to block or otherwise constrain lesser known search engines, either because they are not so well-behaved, they do not support the REP, or because we simply do not know what they actually support.

Finally, there are a couple of points to consider when it comes to using the REP. As mentioned before, robots can choose to ignore the robots.txt if they so wish, so it is by no means a guarantee that our content will be left alone by robots. Also, the robots.txt is a publicly available file, meaning that anyone can see which directories we do not want robots to crawl. This makes it a bad idea to list directories in the robots.txt which we really do not want anyone else to know about or see.

It is also important to remember that the REP is not suitable for controlling access to private content. However, if used in conjunction with proper authentication mechanisms, it can add an additional level of security. For example, if a situation should occur where sensitive personal data is made publicly available online by a mistake (caused by human error or otherwise),

[8] `http://www.w3.org/TR/html4/appendix/notes.html\#h-B.4.1.1`

the potential damages could be greatly reduced if directives are already in place telling search engines not to index or crawl that content.

## 4.6 Summary

In this chapter we have looked at the only method available to website owners for providing search engine robots with crawling and indexing instructions - the Robots Exclusion Protocol. We have also conducted an experiment with the aim of evaluating how reliable this method is. In the next chapter we take a look at the WordPress platform, and investigate to what degree it allows publishers to hide personal data from search engines.

# Chapter 5

# WordPress and Privacy

In this chapter we conduct a small study on the privacy aspect of the WordPress platform, in relation to search engine visibility. First we will present the privacy options offered by WordPress. We then take a look at how the plugin architecture of WordPress allows us to create solutions for increasing the privacy aspect of the platform. In Section 5.4 we present some of the privacy related plugins that are available to WordPress users from the WordPress Plugin Directory. Finally, in Section 5.5, we discuss the privacy aspect of WordPress and how its plugin architecture allows developers to improve it.

## 5.1 The WordPress Platform

WordPress is a free, open source publishing platform, based on PHP and MySQL. It started out as just a blogging system, but has evolved into a full-scale content management system [55]. It is currently the largest self-hosted blogging tool, with a CMS market share of 53.7%, according to W3Techs [56]. Note that this thesis is concerned with the blogging software found at *Wordpress.org*, and not the blog web hosting service provider found at *Wordpress.com*.

## 5.2 Search Engine Visibility

As of Version 3.5.0[1], the WordPress software offers users a search engine visibility option (see Figure 5.1). According to the WordPress Codex[57], the following happens when selecting this option:

- WordPress generates a "noindex,nofollow" *robots meta tag* in the `<head></head>` section of the site's source.

[1] `http://codex.wordpress.org/Version_3.5`

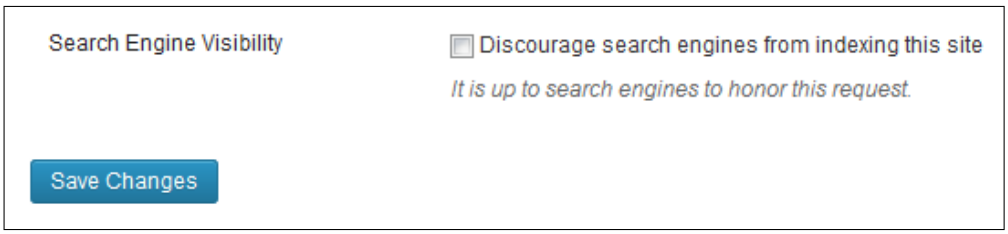


Figure 5.1: Privacy options for sites created with the wordpress.org software

- Causes hits to *robots.txt* to send back (provided that no such file already exists):
  ```
  User-agent: *
  Disallow: /
  ```
- WordPress stops pings to any RPC ping services specified in the UpdateServices-option in the administration panel.
- Hides the Update Services option in the Administration Settings menu and replaces it with the message "WordPress is not notifying any Update Services because of your blog's privacy settings."

As we can see in Figure 5.1, the text next to the checkbox states that ticking off the box will "discourage search engines from indexing this site". However, as we learned from our study on the Robots Exclusion Protocol in Chapter 4, it is not recommended to use the *robots.txt* for discouraging search engines from indexing a site. What the directive does is to tell all search engine robots that they are not allowed to crawl any of the content on the site. But as our experiment showed, pages that have been *disallowed* in a robots.txt may still show up in the search engine result pages, as a URL-only link. Google and Bing also mentions this on their support pages, stating that there is no guarantee that pages will not still show up in the result pages of a search engine, despite the search engines being disallowed from crawling the pages in question. The addition of a *noindex* robots meta tag will not make any difference in this regard, due to the fact that search engine robots will not be able to actually see the directive. Therefore, in order to discourage search engines from indexing content, it is better to only use the *noindex* robots meta tag.

## 5.3 Plugin Architecture

Because WordPress features a plugin architecture, users have the option of extending its abilities beyond the features that are part of the base install. In computing, a plugin (or plug-in) is a set of software components that

adds additional functionality to a larger software application. As stated in the WordPress Codex[2]:

> "WordPress Plugins are composed of php scripts that extend the functionality of WordPress. They offer new additions to your blog that either enhance features that were already available or add otherwise unavailable new features to your site." [58]

The WordPress Plugin Directory[3] currently contains over 24.000 plugins, which are available for anyone to download. But plugins downloaded from elsewhere can also be installed on a WordPress site.

## 5.4 Privacy-Related WordPress Plugins

In this section we will take a look at some of the privacy-related plugins that are available to WordPress users from the WordPress Plugin Directory. By searching for the term *privacy*, we got a result set of 298 plugins that matched the search term. Examining these results reveal that the term privacy covers a wide range of meanings. For example, some of them simply displays a privacy policy upon creating a new user account, while others are targeted towards disabling buttons for sharing content with social networks.

However, there are some that are more related to the focus area this thesis is concerned with. A recurring functionality that several of them seems to offer, is the ability to block access on a site level for anyone who are either not logged in to the site, or who are not authorized users. For example, according to its description, the *Restricted Site Access*[4] plugin allows the site administrator to limit access to visitors who are logged in or are accessing the site from a set of specified IP addresses.

In addition, some of the plugins seems to give a bit more fine-grained access control, by allowing the administrator to block access to certain content (e.g., a post tagged as private, or specific parts of the content within a post), and specify that only some members or user groups are allowed to view the blocked content. Examples include the *Restrict Lite*[5] plugin, which, according to its description, allows the publisher to use so called shortcode[6] to limit what data is shown to users. By placing certain content inside the shortcode tags, it will be restricted to logged in users with the correct capabilities.

None of the plugins mentioned above are specifically targeted towards search engine robots and search engine visibility, so we also did a search for the terms *robots* and *crawlers*. This yielded result sets of 138 plugins and

[2]The WordPress Codex is the official WordPress documentation
[3]`http://wordpress.org/extend/plugins/`
[4]`http://wordpress.org/extend/plugins/restricted-site-access/`
[5]`http://wordpress.org/extend/plugins/restrict-lite/`
[6]`http://codex.wordpress.org/Shortcode`

Table 5.1: Privacy-related plugins for WordPress

| **Category** | **Number of plugins** | **Percentage (out of 514)** |
|---|---|---|
| **People access control** | 19 | 3.69% |
| **Content-based restriction** | 5 | 0.97% |
| **Robot-oriented restriction** | 22 | 4.28% |
| **None of the above** | 468 | 91.05% |

78 plugins respectively. Among these, we found several that allows the site administrator to add additional directives to the robots.txt. A good example here is the *PC Robots.txt*[7] plugin, which generates a virtual robots.txt file, and then allows the user to edit its content from the plugin settings page. There are also some plugins that provides the ability to add robot META tags for each post or page on a site. The *WordPress Meta Robots*[8] plugin, for example, provides the publisher with a dropdown menu where they can select between four different combinations of robot META tags, like *index, nofollow* and *noindex, nofollow*.

In order to get an indication of how many of the plugins available for download from the WordPress Plugin Directory are related to the focus area this thesis is concerned with, we divided them into four categories, and counted how many fell into each category. In the cases where a plugin fell into more than one category, it was counted more than once. The categories were: (1) Those concerned with controlling access for people, (2) those concerned with restricting access based on the content, (3) those concerned with robot-oriented restriction, and (4) everything else. Note that this count is only a rough calculation, based on a quick reading of the plugin descriptions. Several of the descriptions provided too little information to understand what they actually did. Some of them were also written in languages other than English. And there may be some overlap of the search results between the search terms used. In addition, some plugins seems to be included more than once. This means that the total number of plugins is somewhat misleading. The count is presented in Table 5.1.

The solution developed as part of this thesis falls into a combination of the categories *Content-based restriction* and *Robot-oriented restriction* from Table 5.1. Out of all the plugins we searched through, only four of them seemed to fall into roughly the same combination of categories as our solution. These

[7] `http://wordpress.org/extend/plugins/pc-robotstxt/`
[8] `http://wordpress.org/extend/plugins/wordpress-meta-robots/`

were the *Enigma*[9], the *Image Text*[10], the *Email Address Encoder*[11], and the *PHPEnkoder*[12]. The purpose of the *Enigma* plugin is, according to its description, to encrypt text in order to avoid email addresses and any other sensitive content to be collected by robots. What it does is to allow the user to place content they do not want search engines to see inside shortcode, on the form `[enigma]...[/enigma]`. From looking at the source, it seems to rely on the fact that search engines have typically had difficulty reading JavaScript generated content, and so any content inside the [enigma] tags should then not be visible to search engine robots. The *PHPEnkoder* seems to function in roughly the same way, while the *Email Address Encoder* works by encoding email addresses into decimal and hexadecimal entities. The *Image Text* plugin works, according to its description, by converting text into images. What these four plugins have in common is that they attempt to prevent robots from viewing specific types of content, while still allowing normal users to see it.

## 5.5 Discussion

The default privacy settings offered by WordPress does not allow for a very fine-grained level of control over any of the content that is published. In the context of search engines visibility, WordPress users have to choose between being having their site (with all its content) completely visible to search engines, or not at all. Given the important role search engines are likely to play in directing traffic to a site, most users probably do not want to block them from indexing their site. But this only leaves them with the option of letting search engines have free reign over their published content, without any way of influencing how this content should be indexed. Fortunately, WordPress features a plugin architecture, allowing users to implement additional functionality, such as increased privacy protection. Users can either create these plugins themselves, or hope that someone else has done it for them.

In Section 5.4 we looked at the privacy related plugins that are available in the WordPress Plugin Directory. Most of these plugins are aimed at hiding content from other users, either by redacting some parts of the content, or by blocking entire posts or pages. There were also some plugins that focused on search engine visibility, either by adding the ability to generate a robots.txt file and edit it, or by adding robot META tags. This way, users can tell well-behaved robots which parts of the site they are not allowed to crawl or index.

As part of this thesis, we have created a WordPress plugin that addresses

[9] `http://wordpress.org/extend/plugins/enigma/`
[10] `http://wordpress.org/extend/plugins/imagetext/`
[11] `http://wordpress.org/extend/plugins/email-address-encoder/`
[12] `http://wordpress.org/extend/plugins/php-enkoder/`

the limited privacy options that exists in the default WordPress installation. Apart from four other plugins, our solution differs from the already existing privacy related plugins in that it focuses on giving publishers more fine-grained control over what content search engines are allowed to index, without blocking access to any human visitors who try to view the content. The four other plugins mentioned previously seems to offer similar functionality to ours, by allowing the user to restrict access for robots to certain content. Where they differ is that they are not specifically aimed at personal names. They also offer far less functionality. For example, they do not appear to allow the user to choose whether some robots should be exempt.

## 5.6 Summary

In this Chapter we have looked at the privacy options offered by WordPress. We have also looked at some of the privacy-related plugins developed by the WordPress community. In the next chapter, we present our solution, in the form of a WordPress plugin, that is aimed at improve the privacy aspect of WordPress, as well as increase the control WordPress users have over personal data published online.

# Chapter 6

# A Privacy Enhancing WordPress Plugin

So far in this thesis we have talked about the reasons for why it is important to have control over personal data published online. In Chapter 4 we saw how robot directives can be used in this regard, by preventing search engines from indexing specific pages and copying the content into their database. However, search engines play an important role in directing traffic to a website, so completely blocking a search engine robot from accessing their content may not be what a website owner wants. We also saw, in Chapter 5, that the WordPress platform does very little to help publishers hide personal data from search engines.

In Chapter 1 we specified that one of our research objectives would be to develop a tool that would facilitate increased control over personal data published online. In this chapter we present our solution in the form of a plugin which can be deployed on any WordPress website. Its purpose is to allow publishers using the WordPress platform to redact personal names from content that is viewed by search engine robots, with the aim of avoiding personal names being indexed along with any contextual data attached to them. The plugin works by checking whether the visitor to the site is human or a search engine robot. If the visitor is a search engine robot, the plugin will redact any personal names before delivering the content, replacing them with the text [redacted]. To human visitors, the names will appear as normal.

The chapter is organized as follows: In Section 6.1 we specify a set of objectives that the plugin should achieve. Then, in Section 6.2, we explain how these objectives have been met in the form of the functionality offered by the plugin. Afterwards, in Section 6.3, we take a look at how WordPress plugin projects can be published and maintained. Finally, in Section 6.4, we give a brief description of how users can interact with the plugin.

## 6.1 Objectives

Before beginning the task of implementing our solution, we needed to decide on a set of objectives (or requirements) describing what functionality the solution should support. The overall goal was to redact personal names from content that is viewed by search engine robots, while allowing other visitors to see the complete content, including personal names. This meant that the application needed to be able to tell the difference between humans and robots. In addition to the application being able to automatically detect a name and redact it, we wanted to give the user complete control over which names search engines should be allowed to index. This could be achieved by allowing the user to tag a name, or by adding it to a list of names, that should be redacted if the content is viewed by a search engine robot. Based on the above, we decided on the following set of objectives that we wanted our solution to achieve:

- It should be able to detect if a search engine robot tries to view the content.
- It should allow the user to add additional search engine robot names to the list that the application checks up against when deciding if a visitor is a human or a robot.
- It should allow the user to manually tag a name.
- It should be able to automatically identify and tag personal names, without the user having to do it manually.
- It should allow the user to specify an opt-in list of names that should always be tagged, regardless of whether or not the application identifies it as a personal name.
- It should allow the user to specify an opt-out list of names that should not be tagged, regardless of whether or not the application identifies it as a personal name.

## 6.2 Implementation

We briefly mentioned in Chapter 5 that WordPress features a plugin architecture. This allows users to extend the ability of the platform beyond the features that are part of the core installation. Taking advantage of this, we have implemented our solution as a plugin for WordPress. The plugin has been given the name “Name Redactor”, chosen because of its connection to the act of redacting something. According to the Merriam-Webster Dictionary [59], to redact something means:

- "to select or adapt (as by obscuring or removing sensitive information) for publication or release".

- "to obscure or remove (text) from a document prior to publication or release".

In this case, the redacted element will be personal names. The one doing the redaction is (again according to the Merriam-Webster Dictionary [60]) known as the *redactor*, hence the name of the plugin.

This section will describe how the functionality specified in Section 6.1 has been implemented. First we explain how the WordPress API can be used to simplify the process of integrating externally developed software, and how this API has been utilized in the development of our plugin. Next follows an explanation of how the application is able to detect search engine robots, and how personal names are then redacted. The functionality for automatically identifying personal names are also explained.

### 6.2.1 WordPress APIs

In order to facilitate the extension of the WordPress application through the creation of themes, widgets, and plugins, WordPress provides developers with an Application Programming Interface (API)[1]. This is basically a set of functions that simplifies the process of integrating externally developed software, such as plugins. By using the functionality provided by the API, we, as developers, ensure that we are communicating with the core WordPress application in the proper way, that we follow platform best practices, and that the data we operate on is sanitized by using the same mechanisms as that of the core WordPress application. This helps make plugins more secure. It also makes them more resilient, in that whenever the core application is updated or otherwise changed, the standard functions will have to go through the usual deprecation process, giving the plugin developer time to upgrade the application before it is no longer compatible with the rest of the platform. Another advantage of using the functions provided by the API, is that developers does not have to waste any time creating their own functions, and run the risk of breaking something in the WordPress core installation.

The WordPress API can be separated into several different API topics, with each covering a set of functions involved with a specific functionality. Below we will look at the APIs that have been relevant for the implementation of our plugin.

[1] `http://codex.wordpress.org/WordPress_APIs`

### Plugin API

The Plugin API documents the so called *hooks* available to WordPress plugin developers, as well as information on how to use them. These hooks allow a plugin to "hook into" the rest of WordPress, giving it access to various functions within the WordPress core. According to the WordPress Codex[2], there are two kinds of hooks:

1. Actions: Actions are hooks that the WordPress core launches at specific points during execution, or when specific events occur. Using the Action API, a plugin can specify that one or more functions are executed whenever these hooks are launched.

2. Filters: Filters are hooks that the WordPress core launches in order to modify various text content types before adding it to the database or sending it to the requester (e.g. a user's browser). Using the Filter API, a plugin can specify that certain functions are executed whenever these hooks are launched, in order to modify specific types of text before passing it on.

Our plugin makes extensive use of several of these hooks. For example, in order to redact specific parts of the content before allowing a search engine robot to see it, we use the filter hook *the_ content*[3]. This filter is used to filter the content of a post or page after it is retrieved from the database and before it is printed to the screen. This allows our plugin to make changes to the content (i.e., redact personal names) before presenting it to any visiting search engine robots.

### Quicktags API

One of the requirements we established at the beginning of the design process, was that the user should have the ability to manually tag a name in pages, posts and comments. The way we have implemented this functionality, is to add a *redact button* in the Text editor of WordPress (see Figure 6.4 on page 60). This allows the user, while writing some content, to easily select a name and click the redact button in order to tag it. The redact button has been created using the Quicktags API[4], which allows developers to include additional buttons in the WordPress Text editor.

### Settings API

The Settings API[5] simplifies the process of creating menus and settings pages (or add additional content to already existing pages), as well as the saving,

[2] `http://codex.wordpress.org/Plugin_API`
[3] `http://codex.wordpress.org/Plugin_API/Filter_Reference/the_content`
[4] `https://codex.wordpress.org/Quicktags_API`
[5] `https://codex.wordpress.org/Settings_API`

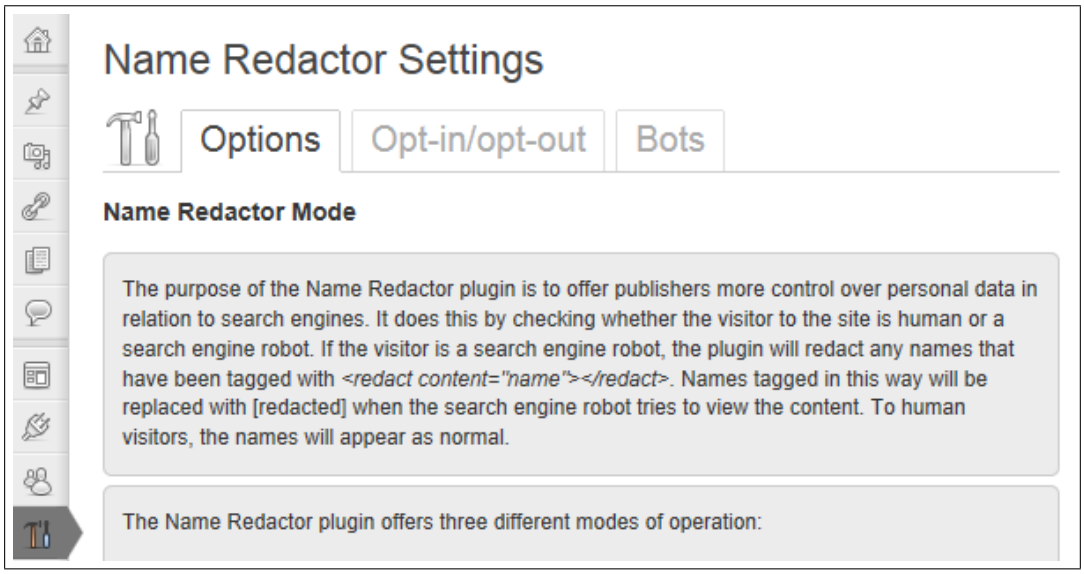


Figure 6.1: Screenshot of (part of) the plugin settings screen.

validating, and retrieving of user input.

In order to let the user customize how the plugin functions, through editing option values and entries in the database tables, the plugin offers its own settings page (Figure 6.1). This settings page (or settings screen) comes in the form of an administration menu item located in the main navigation menu, allowing the user to access it from any of the Administration Screens. While the settings screen was built using the Settings API, the function that generates the settings screen was registered using the Plugin API. According to the WordPress Codex, "a top-level menu displays as new section in the administration menus and contains sub-level menu items. A sub-level menu means the menu item is a member of an existing menu" [61]. As our plugin did not introduce an entirely new concept or feature to WordPress, it was decided that the plugin did not warrant a new top-level menu. Instead, it was placed below the top-level menu *Tools* as a sub-level menu.

Instead of creating a new settings page for each group of options offered by the plugin, the settings page uses tabbed navigation, grouping all the different sets of related options into a single page.

### 6.2.2 Storing Data

Our plugin uses a built-in mechanism[6] in WordPress that allows us to save, update, and retrieve individual, named pieces of data (or "options") in the WordPress database. In addition, the plugin creates two database tables

[6] `http://codex.wordpress.org/Writing_a_Plugin#WordPress_Options_Mechanism`

upon activation that are also stored in the WordPress database. These tables contain the names of search engine robots and the names specified in the opt-in and opt-out lists. According to the WordPress Codex, creating new, custom database tables is "appropriate for data not associated with individual posts, pages, attachments, or comments – the type of data that will grow as time goes on, and that doesn't have individual names" [62].

### 6.2.3 Detecting Search Engine Robots

The first step in the implementation of this plugin was to device a way for the plugin to detect if a visitor to the site was a human or a search engine robot. As explained in Section 2.3 on page 14, web crawlers typically identify themselves to a web server using the User-agent field of a HTTP request. The User-agent string format is specified by section 14.43 of the RFC 2616 (HTTP/1.1) [30]. This allows our plugin to check if a visitor to a site is human or a search engine robot. For example, if Googlebot[7] requests to view a page from a web server, that request will be identifiable by a user-agent string containing "Googlebot" and the host address `http://www.google.com/bot.html`. According to the Google support pages [63], the Googlebot user-agent string would look like this:

```
Mozilla/5.0 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html)
```

or (rarely used):

```
Googlebot/2.1 (+http://www.google.com/bot.html)
```

With this information, an application can the check the user-agent string up against either a list of names of various search engine robots (e.g., "Googlebot"), or a list containing the whole user-agent string of various search engine robots (e.g., like the Googlebot user-agent string above). Of these two methods, our solution user the former, which is more efficient because it does not have to consider multiple user-agent strings for each robot (e.g., Google uses several different robots, each with their own user-agent [63]). It is also more reliable, in that as long as the search engine robot does not change its name, it does not matter if its user-agent string should change. So for each search engine robot name in the list, it checks if that name occurs within the user-agent string, and if it does, redacts the tagged content. The plugin has a default set of 13 search engine robot names, including Googlebot and Bingbot[8], with the user having the option of adding more in the plugin settings menu (Figure 6.2 on the following page). Additional names can for example be found in the Web Robots Database[9], which lists many common robots.

[7] The search bot software used by Google
[8] The search bot software used by Bing
[9] `http://www.robotstxt.org/db.html`

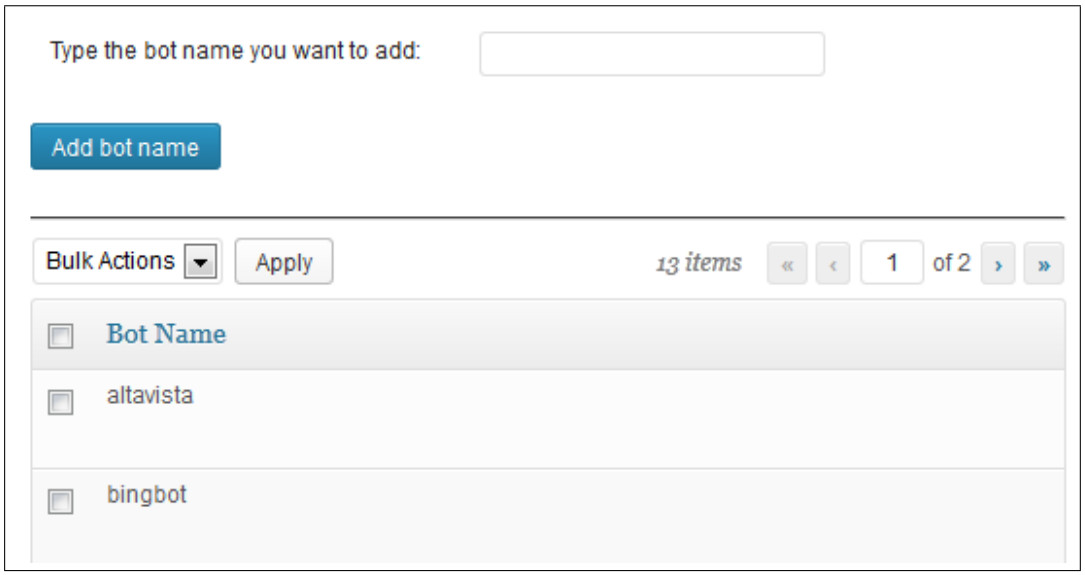


Figure 6.2: Screenshot of the Bots section, where the admin can maintain a list of bot names for the plugin to check up against.

### 6.2.4 Tagging Names for Redaction

Personal names are tagged by inserting "XML-like" tags around the name, on the form `<redact content="name"></redact>`. If the application detects a search engine robot, it will retrieve the content from the database, search for any names tagged for redaction, and replace anything between the opening tag `<redact content="name">` and closing tag `</redact>` with the text *[redacted]*, before passing the content on to the robot. These tags are not visible to human visitors, unless they were to look at the source code. For example, if a user was to tag the name "Eliza", the source code would look like this: `<redact content="name">`Eliza`</redact>`. To a human visitor, the name would appear as normal, whilst to a search engine robot, it would appear as `<redact content="name">`[redacted]`</redact>`.

### 6.2.5 Automatic Name Detection

In addition to manually tagging personal names, the application also has the ability to automatically identify words as names and redact them accordingly. However, this function is not completely accurate, and is primarily meant as a supplement to manual tagging.

One problem with implementing a function like this is how to define what constitutes a personal name. For European languages it may be sufficient to only consider words which begin with a capital letter. However, this will include words that are not personal names, such as Microsoft and London.

In addition, it would also include every first word in every sentence (which normally begins with a capital letter).

For our application, personal names have been defined as one or more consecutive words with their first letter capitalized, unless the word is at the beginning of a sentence. This definition will cause some words to be identified as personal names, even though they are not, and cause some words that are personal names, to not be identified as such. Our definition of a personal name has been implemented as two different regular expressions. A regular expression can be said to be a string that can be used for describing a search pattern. These regular expressions are shown below:

- `/(?<!\.\s|^)[A-Z][a-z]+(?=\s[A-Z])(?:\s[A-Z][a-z]+)*/m`
- `/(?<!\.\s|^)\b[A-Z][a-z]+\b/m`

The first regular expression matches two or more consecutive words starting with a capital letter, unless the first word is at the beginning of a sentence. The second regular expression matches one word with the first letter capitalized, unless that word is at the beginning of a sentence. Note that if the first word of a sentence is preceded by more than one space, that word will be matched as well. The reason for this is that the negative lookbehind[10] construct for PHP, used for checking that the word is not at the beginning of a sentence, does not support an unknown number of spaces. Additionally, if the first word of a sentence is directly preceded by a character (e.g., a quotation mark) it will identify that word as a personal name, provided that its first letter is capitalized. This highlights one of the weaknesses of using regular expressions for this task, as it can be extremely difficult to try to cover every eventuality.

### 6.2.6 Programming Language and Tools

The program has been created using PHP[11] for writing most of the code, JavaScript for creating a redact button in the WordPress Text editor, MySQL[12] for creating the database tables used by the plugin, and HTML[13] for creating user-input forms and structuring text. The plugin tries to follow the WordPress coding standards[14] as much as possible.

The program code was written using the Open Source software editor *Notepad++*[15]. Its use is governed by the GPL Licence[16].

[10] `http://www.regular-expressions.info/lookaround.html`
[11] `http://php.net/`
[12] `http://www.mysql.com/`
[13] `http://www.w3.org/TR/REC-html40/`
[14] `https://codex.wordpress.org/WordPress_Coding_Standards`
[15] `http://notepad-plus-plus.org/`
[16] `http://www.gnu.org/copyleft/gpl.html`

In order to continually be able to run and test the code, a virtual server was set up that could be used to create and host a local WordPress website. For this purpose we used the free edition of a software called *Desktop Server*[17]. The virtual server resides locally on the computer, and can be accessed through a web browser via a fictitious, pseudo top-level domain name (e.g., `http://www.example.dev`). The WordPress site created using this software worked in mostly the same manner as a normal WordPress site hosted online would.

### 6.2.7 License and Availability

Our solution has been released under the GPL license (version 2 or later)[18] from the Free Software Foundation[19], which is also the license under which the WordPress software is released. This was also a requirement in order to get it hosted in the Wordpress Plugins Directory. This means that the program is free software, and anyone can freely redistribute it and/or modify it under the terms of the GNU General Public License, version 2, as published by the Free Software Foundation.

## 6.3 Maintaining a WordPress Project

One of the considerations WordPress plugin developers have to make is how the plugin should be published and how to maintain it. One option is to host in on their own website. The other option is to set up a WordPress plugin repository at the wordpress.org website and maintain the project there. The wordpress.org website offers to host plugins, as long as they adhere to certain rules [64]. For example, the plugin must be compatible with the GNU General Public License v2, and it must not do anything illegal or be morally offensive. For a WordPress plugin developer, there are several benefits of having their plugin hosted in the WordPress Plugin Directory, some of which are listed below:

- The plugin becomes searchable and downloadable to all Wordpress users, not only through the Wordpress site, but also from within the WordPress Dashboard.

- Whenever a new version of the plugin is uploaded to the Directory, users who have it installed in their WordPress will get a notification saying that a new version is available, and be offered the option of updating their current version.

[17] `http://serverpress.com/products/desktopserver/`
[18] `http://www.gnu.org/licenses/gpl-2.0.html`
[19] `http://www.fsf.org/`

- The developer can track basic statistics regarding how many times their plugin has been downloaded.
- It gives the developer a centralized place where users can ask questions and provide feedback in the form of comments, reviews and ratings.
- The rating system allows the plugin to be rated against other WordPress plugins.
- It gives a plugin greater exposure to the WordPress community.

An added benefit is that plugins hosted in the WordPress Plugin Directory may be perceived as being more trustworthy than plugins hosted elsewhere, perhaps by making it seem more "official". Certainly a plugin with a high number of downloads coupled with a high rating score will add to the trustworthiness of it. According to the Wordpress support pages, plugins submitted to the WordPress Plugin Directory will be manually reviewed before either being approved or rejected. However, it remains unclear exactly what this entails, and if it involves looking for security holes or checking the quality of the plugin. The WordPress Codex states that "WordPress Plugins hosted in the WordPress Plugins Directory are considered thoroughly tested and "safe"" [58], but it also states that "WordPress Plugins are the responsibility of the author and the user, and they are typically works-in-progress as WordPress grows and expands" [58]. Considering the time it took for our plugin to be approved (see below), it is doubtful whether this involves a very thorough analysis.

### 6.3.1 Getting the Plugin Published

Upon completing the plugin, it was decided that we should attempt to have our plugin hosted by WordPress. The first step in this process was to register a user account at wordpress.org. Next we needed to fill out a form, where we described the plugin and added a link to a ZIP file of a working version of the plugin. In addition to the plugin itself, the ZIP file also included a readme file containing detailed instructions for installation and usage, as well as explaining the purpose of the plugin. We then had to wait for it to be approved by the WordPress plugin review team. WordPress does not specify how long this approval process is expected to take, only that: "within some vaguely defined amount of time, your plugin will be manually reviewed. You may be emailed and asked to provide more information" [64]. In our case, it took less than a day for the plugin to be approved. After the plugin had been manually reviewed and approved, we were given access to a Subversion Repository[20] on their servers, where the plugin files would be stored. From this repository, anyone can *check out* a copy of the plugin files onto their

[20] `http://subversion.apache.org/`

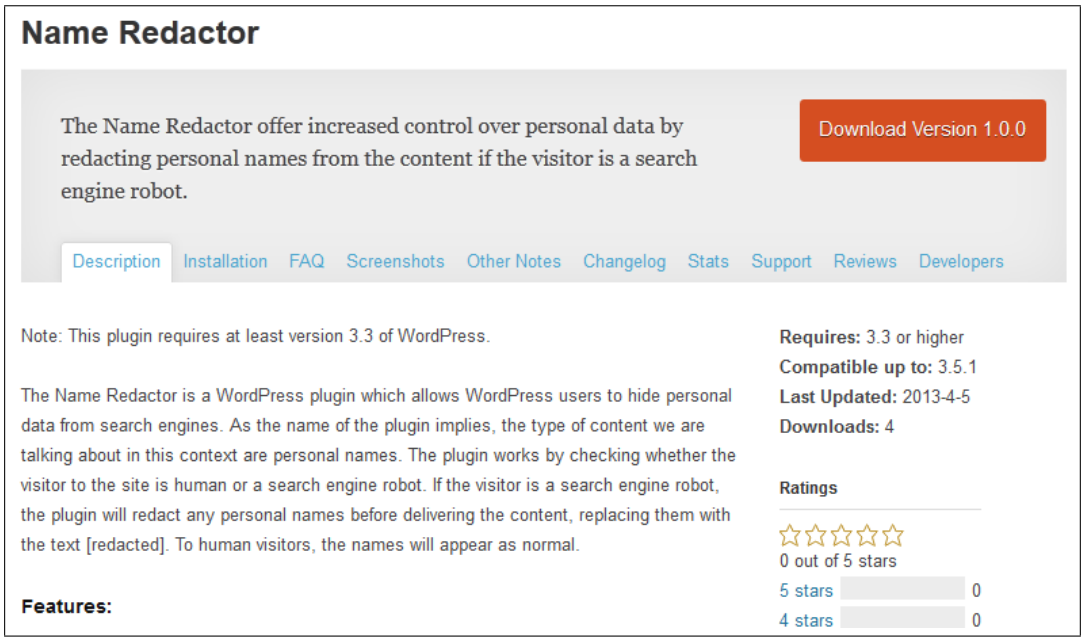


Figure 6.3: Version 1.0.0 of the Name Redactor listed in the WordPress Plugins Directory.

local computer, but only the plugin author can *check in* files, i.e., make changes to the files, add new files, and delete files. Once the plugin had been added to the WordPress Plugin Directory, users would be able to find the plugin by searching for it. In order to make it easier to find, the plugin was tagged with the following tags: comments, hide, names, page, posts, privacy, redact, redaction. Figure 6.3 shows a screenshot of how version 1.0.0 of the Name Redactor appears in the WordPress Plugins Directory. In addition to providing a download button, the listing is divided into several sections, like *Description*, *FAQ*, and *Support*. These sections may provide additional information, or in the case of the latter section, allow users to receive support by posting any questions they might have.

## 6.4 Interacting With the Plugin

This section will give a brief description of how users can interact with the plugin. As soon as the plugin has been installed and activated on a WordPress site, the user can begin creating content and manually tag personal names occurring within the content. Any adjustments to the way the plugin behaves can be made in the plugin settings screen. Both of these aspects of using the plugin will briefly be described below.

### 6.4.1 Tagging Personal Names

When writing a new blog post, the user has the option of either using the *Visual* or *Text* mode of the editor. The Visual mode lets the user see the post as it will look like when it is published, while the Text mode shows the code, such as `<strong></strong>` for strong emphasis of text, in addition to the normal text. While in the Text editor, the user has the option of using so called *quicktags*, located above the post editing area. These can be inserted into the text either by clicking a button for inserting an opening tag, write some text, and then clicking the same button to insert a closing tag, or by selecting a portion of the text and then clicking a button for enclosing the text with the selected HTML tag. Our plugin supports displaying an additional quicktag, called the *redact button*, which works in the same way as the others. This means that the user can quickly tag personal names while writing. In WordPress, when the user is ready to publish the content they have written, they have the option of previewing it first in order to see how it will appear when published. In the plugin settings screen, the user can choose that the preview mode should display the content as it would look like from a search engine robot's point of view (i.e., the names that have been tagged — either manually or automatically — will be replaced with the text [redacted]. Figure 6.4 on the following page illustrates how two personal names have been manually tagged using the redact button located in the WordPress Text editor.

### 6.4.2 Adjusting the Plugin Settings

The plugin settings screen allows the user to customize how the plugin functions. Here, the user can choose the type of content the plugin should redact tagged names from (i.e., posts, pages and comments), and whether the plugin should check up against an opt-in or opt-out list. This has been implemented as a single list where the user can add additional names, or remove existing names (Figure 6.5 on page 61). When adding a new name, it will be labeled as either opt-in or opt-out. Names labeled as opt-in will be tagged, while names labeled as opt-out will not (unless they have been manually tagged). The user can also add additional robot names that the plugin should check up against. The settings screen also allows the user to choose between the three modes of operation offered by the plugin. The first mode is basically an inactive mode, where the plugin will not attempt to redact any content. In the second mode, the plugin will redact any names that have been tagged manually. In the third mode, in addition to redacting manually tagged names, the plugin will attempt to identify personal names and redact them as well, as explained in Section 6.2.5.

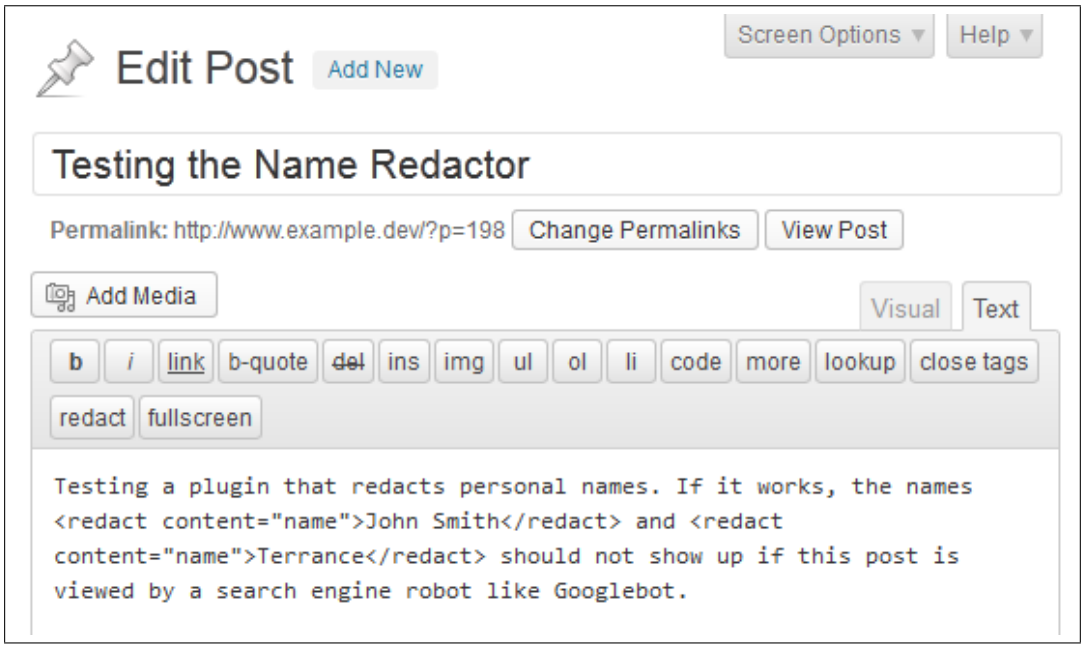


Figure 6.4: Screenshot of how two personal names — John Smith and Terrance — have been manually tagged in the text, using the redact button.

## 6.5 Summary

In this chapter we have presented our solution for facilitating increased control over personal data published online. The solution has been implemented as a plugin for the WordPress software, allowing it to be deployed on any WordPress website. In the next chapter, we evaluate our solution.

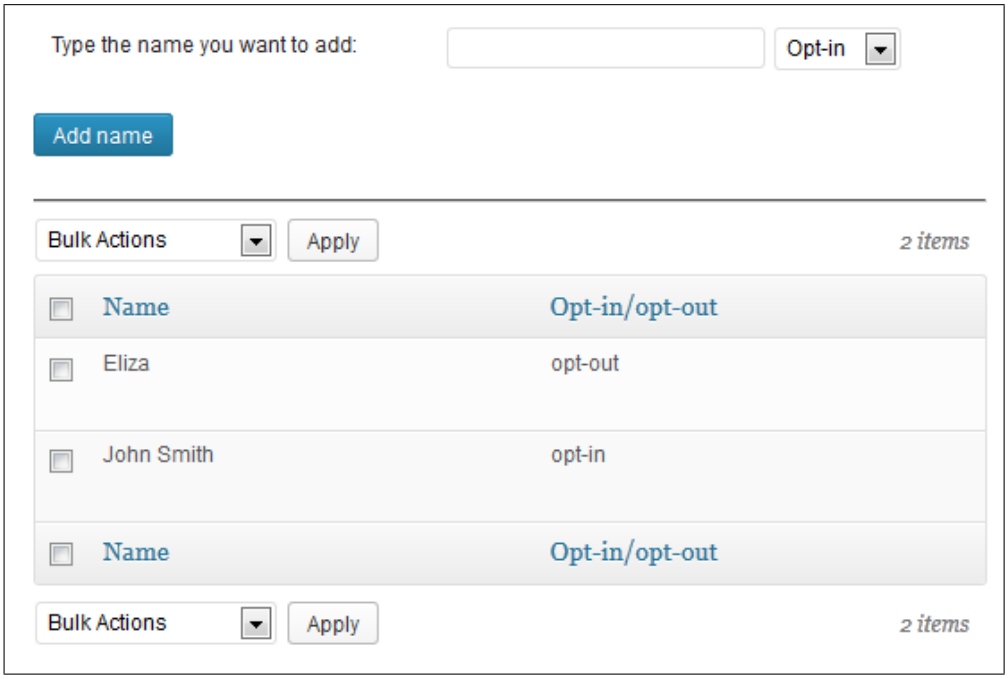


Figure 6.5: Screenshot of the Opt-in/opt-out section, where the admin can maintain lists of opt in/opt out names.

# Chapter 7

# Evaluating the WordPress Plugin

In order to evaluate the solution presented in Chapter 6, we conducted both a software test and a usability test. This chapter describes how the tests were conducted and the methods employed, and then present the findings from those tests. The purpose of the evaluation was to check that the solution fulfilled its intended purpose. The main goal of the software test was to verify that the solution functioned according to the stated objective, which was to be able to separate personal names from the contextual data attached to them when the content is indexed by a search engine. The main goal of the usability test was to check that its intended user group would be able to actually use the solution. In general, the results showed that while the solution functioned as intended, there were some usability aspects that were problematic and possibly needs to be redesigned. The chapter is organized as follows: In Section 7.1 we describe the software test, while the usability test is described in Section 7.2. Each test has its own section were we discuss the results.

## 7.1 Software Testing

This section describes the software test involving the solution presented in Chapter 6. First we start off by clarifying the aim of the study. We then describe how the test was conducted, before presenting the findings in Section 7.1.3. The results from the test are then discussed in Section 7.1.5.

### 7.1.1 Aim of the Study

The aim of this test was to verify that the solution functioned according to the stated objective, which was to be able to separate personal names from the contextual data attached to them when the content is indexed by

a search engine. This aim was formulated as the following question:

- On a Wordpress website with the Name Redactor plugin installed and activated, are personal names indexed along with the rest of the content by search engines, or are they redacted when viewed by search engine robots?

### 7.1.2 Method

The application offers two ways of tagging personal names for redaction. The user can either tag a name manually, or they can let the application attempt to automatically identify and tag a name. In this test we wanted to verify that personal names were redacted correctly in both cases.

In addition, as a supplement to automatic name detection and manual tagging, the user has the option of using an opt-in list and/or opt-out list. Names added to the opt-in list will always be tagged, even if the application does not identify it as a name. Names in the opt-out list will not be automatically tagged, but the user can still manually tag them. In this test we also wanted to verify that names added to the opt-out list were not redacted, as long as they had not been tagged manually.

Based on the above, this test was split into three test cases: **(1)** Check that manually tagged names are redacted correctly, **(2)** check that words automatically identified as personal names are redacted correctly, and **(3)** check that names in the opt-out list are not redacted.

#### Materials

The experimental website was created using version 3.5.1 of the WordPress software[1], and was hosted on a server provided by the University of Oslo[2]. Version 1.0.1 of the solution described in Chapter 6 was then installed on the site used for the experiment. The search engine used to collect the results was Google[3]. In order to change the user-agent of our browser, we used a browser extension called User Agent Switcher[4].

#### Procedure

In order to test our solution, we set up an experimental WordPress website, were we installed and activated our solution. This website was then used to publish some content online in the form of two different blog posts. The content of both blog posts consisted of extracts from the book *Three Men in a Boat* by Jerome K. Jerome. The first blog post would be used for the first

[1] `http://wordpress.org/download/`
[2] `https://www.uio.no/`
[3] `https://www.google.no/`
[4] `http://chrispederick.com/work/user-agent-switcher/`

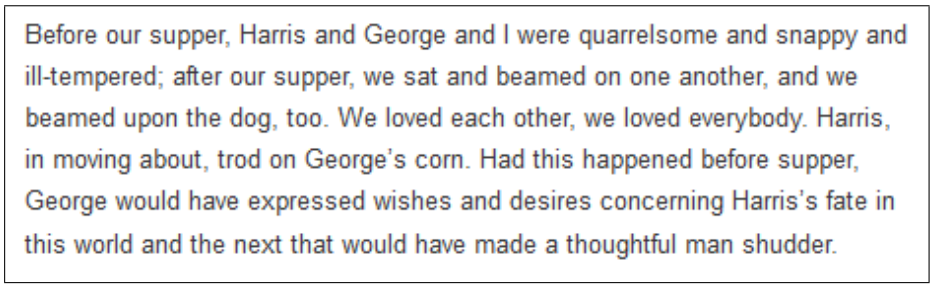
Before our supper, Harris and George and I were quarrelsome and snappy and ill-tempered; after our supper, we sat and beamed on one another, and we beamed upon the dog, too. We loved each other, we loved everybody. Harris, in moving about, trod on George's corn. Had this happened before supper, George would have expressed wishes and desires concerning Harris's fate in this world and the next that would have made a thoughtful man shudder.

Figure 7.1: Screenshot of how the published post appears to a normal visitor to the website.

```
                <div class="entry-content">
            <p>Before our supper, <redact
content="name">Harris</redact> and <redact
content="name">George</redact> and I were quarrelsome and snappy
and ill-tempered; after our supper, we sat and beamed on one
another, and we beamed upon the dog, too. We loved each other,
we loved everybody. <redact content="name">Harris</redact>, in
moving about, trod on <redact content="name">George</redact>'s
corn. Had this happened before supper, <redact
content="name">George</redact> would have expressed wishes and
desires concerning <redact content="name">Harris</redact>'s fate
in this world and the next that would have made a thoughtful man
shudder.</p>
                    </div><!-- .entry-content -->
```

Figure 7.2: The source code of blog post 1.

test case, where both names would be tagged manually, and included two personal names: *George* and *Harris*. Figure 7.1 shows a screenshot of the blog post as it appears to a human visitor. Notice that both personal names are visible. However, if we look at the source code (Figure 7.2), we see that both names have been tagged with `<redact content="name"></redact>`.

After a period of time had passed, we did a search for the website on *Google* and *Bing*. Unfortunately, by this time our experimental website only appeared to have been indexed by Google, and so the results for this test case have only been gathered using the Google search engine[5].

We used three different search queries. The first query was on the form *site:<URL>*, replacing *<URL>* with the actual URL of the website, and was aimed at ensuring that the content had actually been indexed by Google. The second and third queries were on the form *site:<URL> <personal name>*, again replacing *<URL>* with the actual URL of the website. *<personal name>* was replaced with the names *George* and *Harris* for the second and third queries respectively. According to the Google support pages [52], the *site* search operator can be used to help focus your searches. In this case, the

[5] `https://www.google.no/`

```
    <div class="entry-content">
        <p>We had been sitting huddled up in our rugs while
George had been telling me this true story, and on his
finishing it I set to work to wake up Harris with a scull. The
third prod did it: and he turned over on the other side, and
said he would be down in a minute, and that he would have his
lace-up boots. We soon let him know where he was, however, by
the aid of the hitcher, and he sat up suddenly, sending
Montmorency, who had been sleeping the sleep of the just right
on the middle of his chest, sprawling across the boat.</p>
            </div><!-- .entry-content -->
```

Figure 7.3: The source code of blog post 2.

first search query would return any pages belonging to the specified domain, while the second and third queries would return any pages belonging to the specified domain which contained either the name *George*, *Harris*, or both.

The second blog post was used for the remaining two test cases, and included three personal names: *George*, *Harris*, and *Montmorency*. This time, none of the names were redacted manually. Instead, the application settings were changed, so that it would automatically attempt to identify personal names and redact them. If we look at the source code in Figure 7.3, we see that none of the names have been tagged.

This time, instead of waiting for search engines to index our content, we used a browser extension that allowed us to change the user-agent of the browser, from the default to the user-agent of Googlebot. This would trick our application into thinking that we were a search engine robot and act accordingly. We then added the name Montmorency to the opt-out list, because we did not want it to be redacted.

### 7.1.3 Findings

In this section we present our findings from the software testing described in the previous section.

From the listings of results returned by the Google search engine in response to our queries, we were able to determine that the content we had created earlier was not included when the keywords was either the name *George* or *Harris*.

In response to our query that only included the URL of the website, without any of the personal names, the listing of results included the title of the blog post we had created with a reference to the full version, accompanied by a short description (or "snippet") generated from the content of the blog post. The snippet did not contain any of the personal names included in the original blog post. Instead, both names had been replaced with text *[redacted]* (see Figure 7.4 on the next page). The search result also included a cached version of the blog post, and there also we were able to determine

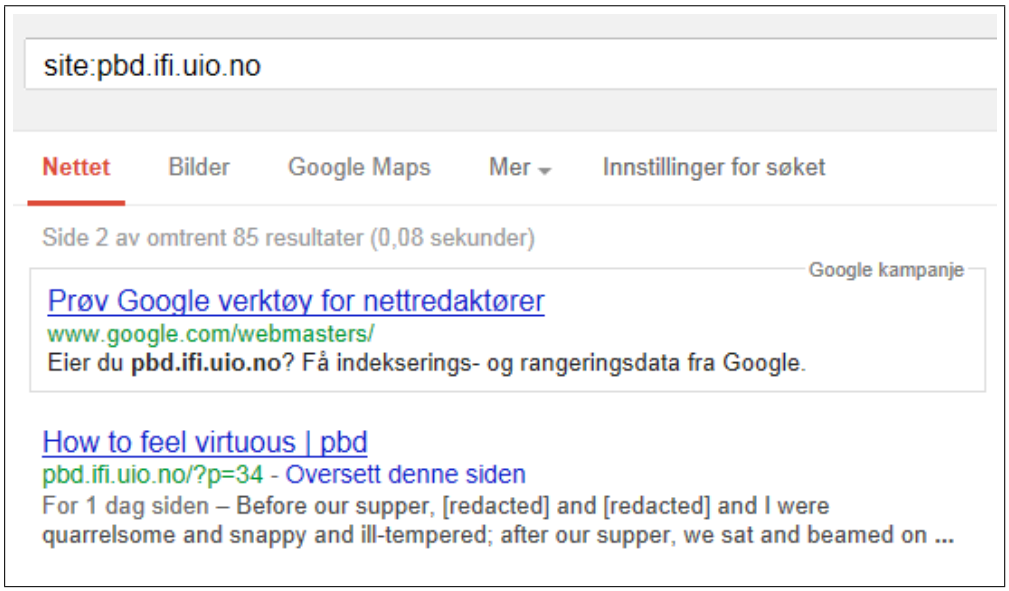


Figure 7.4: Personal names redacted in the Google snippet.

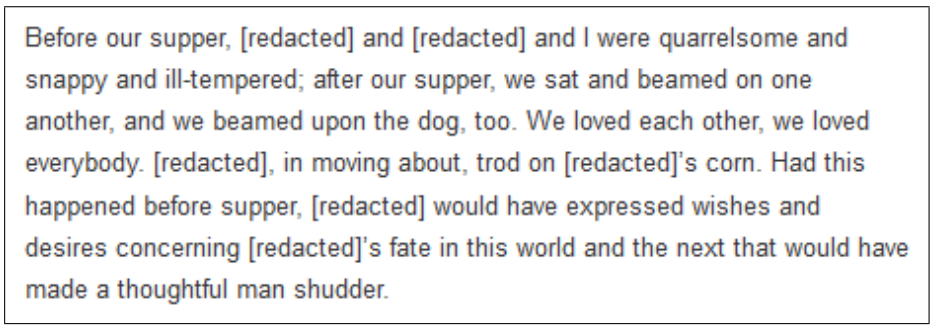


Figure 7.5: Google's cached version of blog post 1.

that all instances of the names George and Harris had been replaced with the text [redacted] (see Figure 7.5).

For the second and third test case, as mentioned previously, we used a browser extension to change our default user-agent into that of the Google search engine robot: Googlebot. From Figure 7.6 on the next page we see that all three personal names have been redacted. However, in addition to redacting the personal names, the application has also redacted the word "we". This suggests that the function responsible for identifying personal names is not 100% accurate. From the source code in Figure 7.7 on the facing page we see how the content would look like to a search engine robot.

Because we did not want the name Montmorency to be automatically

[redacted] had been sitting huddled up in our rugs while [redacted] had been telling me this true story, and on his finishing it I set to work to wake up [redacted] with a scull. The third prod did it: and he turned over on the other side, and said he would be down in a minute, and that he would have his lace-up boots. [redacted] soon let him know where he was, however, by the aid of the hitcher, and he sat up suddenly, sending [redacted], who had been sleeping the sleep of the just right on the middle of his chest, sprawling across the boat.

Figure 7.6: Automatically tagged names have been redacted.

```
                <div class="entry-content">
            <p><redact content="name">[redacted]</redact>
had been sitting huddled up in our rugs while <redact
content="name">[redacted]</redact> had been telling me
this true story, and on his finishing it I set to work to
wake up <redact content="name">[redacted]</redact> with a
scull. The third prod did it: and he turned over on the
other side, and said he would be down in a minute, and
that he would have his lace-up boots. <redact
content="name">[redacted]</redact> soon let him know where
he was, however, by the aid of the hitcher, and he sat up
suddenly, sending <redact content="name">[redacted]
</redact>, who had been sleeping the sleep of the just
right on the middle of his chest, sprawling across the
boat.</p>
                    </div><!-- .entry-content -->
```

Figure 7.7: The source code of blog post 2, with names automatically tagged and redacted.

[redacted] had been sitting huddled up in our rugs while [redacted] had been telling me this true story, and on his finishing it I set to work to wake up [redacted] with a scull. The third prod did it: and he turned over on the other side, and said he would be down in a minute, and that he would have his lace-up boots. [redacted] soon let him know where he was, however, by the aid of the hitcher, and he sat up suddenly, sending Montmorency, who had been sleeping the sleep of the just right on the middle of his chest, sprawling across the boat.

Figure 7.8: The name Montmorency has been opted out.

```
                <div class="entry-content">
              <p><redact content="name">[redacted]
</redact> had been sitting huddled up in our rugs while
<redact content="name">[redacted]</redact> had been
telling me this true story, and on his finishing it I
set to work to wake up <redact content="name">[redacted]
</redact> with a scull. The third prod did it: and he
turned over on the other side, and said he would be
down in a minute, and that he would have his lace-up
boots. <redact content="name">[redacted]</redact> soon
let him know where he was, however, by the aid of the
hitcher, and he sat up suddenly, sending Montmorency,
who had been sleeping the sleep of the just right on
the middle of his chest, sprawling across the boat.</p>
                    </div><!-- .entry-content -->
```

Figure 7.9: The source code of blog post 2, with one of the names not tagged.

redacted, we added it to the opt-out list. This time, when looking at the content (Figure 7.8), we see that while the names George and Harris (as well as the word “we”) are still redacted, the name Montmorency is no longer redacted. The source code (Figure 7.9) also shows that the name Montmorency has not been tagged.

### 7.1.4 Validity Evaluation

Due the fact that Bing did not include our experimental website in their search results, our test results for the server side redaction could only be verified in the case of Google. However, using software for changing our browser user-agent into that of Bing and Yahoo, we were able to duplicate the results (i.e., the Name Redactor correctly redacted the content in the same manner as that of Google), except this time it was on the client side.

This is a strong indication that we could expect to see the same result on the server side for any other search engines which correctly identifies themselves.

### 7.1.5 Discussion

This experiment was aimed at answering the following question:

- On a Wordpress website with the Name Redactor plugin installed and activated, are personal names indexed along with the rest of the content by search engines, or are they redacted when viewed by search engine robots?

Based on our findings, we were able to verify that our solution had managed to prevent Google from indexing any of the personal names that had been manually tagged in the content published on our WordPress website.

In addition, we wanted to check that the function for automatically identifying and redacting names, as well as opting out from this function, worked as anticipated. The results showed that, while the application managed to correctly identify the names, it also identified the word "We" (notice the capital letter) located at the beginning of the first sentence, and then once more later in the post. This suggest that the function responsible for identifying personal names is not completely accurate. One possible reason for the application mistakenly identify the word "we" as a name is that the negative lookbehind[6] construct for PHP, used in the regular expressions responsible for identifying names in our application, does not support an unknown number of spaces. Perhaps the post format of WordPress causes our regular expression to interpret the beginning of the blog post as being preceded by more than one space. This could result in the application interpreting the first word of the first sentence as being the second word. And if that word begins with a capital letter, as the first word in a sentence usually do, it will interpret it as a personal name. The reason for the word "we" being identified as a name later in the post as well, is because once a word has been identified as a name, it will be added to a list of names which will be redacted every time that name occurs in the content.

By using regular expressions to identify personal names, our application will never be able to cover every eventuality. However, as a substitute to manual tagging, it seems to work quite well. It correctly identified and redacted all the occurring personal names, and also correctly did not redact the name specified in the opt-out list.

## 7.2 User Testing

This section describes the user testing involving the solution presented in Chapter 6. First we start off by clarifying the aim of the study. We then

[6] `http://www.regular-expressions.info/lookaround.html`

present the method employed, saying something about the participants chosen for the study, the tasks involved, and the materials used to conduct the test and collect the data. Afterwards we describe how the test was conducted, before presenting the findings in Section 7.2.3. The results from the test are then discussed in Section 7.2.5.

### 7.2.1 Aim of the Study

The usability test was conducted in order to find out whether or not the solution was usable by the intended user group. This could be defined as a person who writes a blog (also called a blogger). More specifically, we were interested in how the participants would fare in terms of:

- Downloading, installing, and activating the application.
- Accomplishing a set of basic tasks by interacting with the user interface offered by the application.
- Understanding the documentation accompanying the application.

### 7.2.2 Method

The usability approach adopted for this evaluation study is *usability testing*, while data gathering techniques used were observation and semi-structured interviews.

The participants were given a set of tasks to complete. While performing the tasks, the participants were encouraged to "think aloud". These tasks included installing the application, navigating through the application settings, and using the application in the context of publishing content. User performance was recorded with the help of software that captured keystrokes and mouse movements. Audio recording was also employed in order to record the "thoughts" of the participant while performing the tasks. Afterwards, a semi-structured interview was conducted.

#### Obligation to Notify

According to the Data Protection Official for Research (NSD):

> "Researchers and students at institutions that have appointed NSD as their Data Protection Official for Research, are to submit their research or quality assurance projects to NSD if the projects are subject to notification according to the Personal Data Act" [65].

This means that if the research involves "collecting/recording/storing personal data by means of a computer, your project is subject to notification. Any information that may be linked to a person is considered personal data"

[65]. As the data generated from the usability testing could not be linked to any person, either directly or indirectly, we concluded that this project was not subject to notification.

### Informed Consent

Before the testing began, the participants were asked to read and sign an informed consent. According to the requirements for informed consent stated by the NSD[7], it described the project and how the usability test would be conducted. It also explained that both audio and screen output would be recorded, as well as specified who would have access to the recorded material, and for how long it would be kept before it would be destroyed. The participants were promised full anonymity. By signing the agreement, the participant affirmed that they had read the agreement and consented to take part in the study.

### Participants

A sample of 5 people were recruited to take part in the test. The number of participants was based on the fact that we had limited time to conduct the study, and it was felt that 5 participants would be sufficient to give us an idea of the usability aspect of the application. In addition, according to [33, p. 647], "it is considered that 5-12 users is an acceptable number to test in a usability study".

Preferably, the participants should have some experience with blogging. Unfortunately, finding people with such experience who were willing to participate in the study proved difficult. Among those who participated, two persons kept a personal Wordpress blog, although one of them used the somewhat limited version hosted at *wordpress.com*, and the other used an older version of WordPress than the one used in this test. However, the findings from this test could give an indication of how new and inexperienced WordPress users would cope with the task of installing and using the plugin. It could perhaps also be argued that if people with little or no experience using WordPress are able to use the plugin, then people with more experience would do equally well, or better.

### Development of the Tasks

The following 8 tasks were developed in order to check how easy or difficult the program was to install and navigate, and how users would complete typical tasks supported by the application. While we could have included more tasks that would cover all the various functions available in the application, it was decided that a smaller set of tasks would be sufficient to give us an

[7] `http://www.nsd.uib.no/personvern/en/notification_duty/consent.html`

idea of the usability aspect of the application's basic functionality, while not being too time-consuming to accomplish during a testing session.

- Task 1: Without leaving the Wordpress Administration Screen, search for the plugin called *Name Redactor*. Once you have found the plugin, you will have the option of reading the documentation, but this is entirely up to you.
- Task 2: Again without leaving the Wordpress Administration Screen, install and activate the plugin.
- Task 3: Write a blog post that contains a personal name. Tag the name for redaction using the plugin, and publish the post.
- Task 4: Change the plugin settings, so that the program automatically tries to identify and tag personal names without you having to do it manually.
- Task 5: Add a name to the list of names that should *always* be tagged automatically.
- Task 6: Add a name to the list of names that should *not* be tagged automatically.
- Task 7: Remove one of the names you just added.
- Task 8: Deactivate and remove the plugin from WordPress.

### Materials

In order to evaluate our solution, a WordPress site was set up, using a software called *Desktop Server*[8], which allows for the creation of virtual servers. Each virtual server resides locally on the computer, and can be accessed through a web browser via a fictitious, pseudo top-level domain name (for example `http://www.example.dev`). This allowed us to quickly create a local Wordpress website that would work just like a normal Wordpress site hosted online. For this study, we used the free Desktop Server Limited edition, which included all the necessary features. In addition to the user testing, the software was also extensively used for software testing during plugin development.

Each participant was then given access to a user account with administrator privileges on the site. This would allow them to install new plugins on the local site, change any of the settings, and create and "publish" new content[9].

[8] `http://serverpress.com/products/desktopserver/`

[9] Any content published on a local site like this will not be accessible to anyone else.

Audio recording and screen capture was achieved through an application running on the computer used for the testing, enabling us to record both audio and screen output simultaneously. Two different applications were tried out. The first application is called *Wondershare DemoCreator*[10], and allows us to capture desktop activities as well as record audio from a microphone. We used a trial version of the application, which had all the necessary functions, but displayed a watermark on the resulting videos. The second application is called *CamStudio*[11], and had most of the same functions as Wondershare DemoCreator. The main difference is that CamStudio is open source software. It also proved to be slightly more difficult to use than Wondershare DemoCreator.

For the interviews, we used a smartphone to record the audio, in addition to taking notes.

### Procdure

The usability testing was conducted individually for each participant. Each testing session lasted for approximately 30 minutes, and consisted of 3 main parts. In the first part we explained the main idea behind the solution, and what we would be evaluating. The participants were then asked whether or not they had any prior experience using the Wordpress platform. Because most of the participants did not have any prior experience using Wordpress, each test session began with a quick introduction to the Wordpress administration screen (called the *Dashboard*), as well as where Wordpress plugins can be found online. This also included how to publish new content. However, no demonstration of how to actually install plugins was given.

In the second part, the participant was given a set of eight tasks to be accomplished. These tasks involved finding and installing the plugin, as well as performing some basic tasks such as changing a few of the plugin settings and manually tagging a name before publishing a blog post. The participants started out in the WordPress Dashboard Screen. This is a tool for quick access to the most used areas of the Administration menu, and can also be used for glimpses into other areas of the WordPress community. The information is presented in blocks, called modules, and these can be added to or rearranged by the user. In this test, we used the default Dashboard setup with no alterations. From there, the participants would have to navigate to different parts of the administration menu in order to accomplish the tasks.

In the third and last part of the session, after the tasks had been completed, a semi-structured interview was conducted with the participant. The interview allowed us to discuss specific situations that arose during the user testing, and gave the participant a chance to ask any questions they might have. We also wanted to collect any thoughts they might have on aspects or

[10] `http://www.wondershare.com/pro/democreator.html`
[11] `http://camstudio.org/`

functions that in their opinion should be changed. The interview was concluded with a question regarding what they thought of the concept of the program, and, had they been blogging, whether or not they thought that they would have used the program.

#### Analysis

For this study we have used a qualitative analysis. For each test session, the audio recording and screen capture from the usability test, along with the feedback from the interview, was analysed by re-listening to the audio recordings and watching the screen captures. The findings from this analysis are presented in Section 7.2.3, and then the most important ones are further discussed in Section 7.2.5.

### 7.2.3 Findings

In this section we present the results from the user testing described in the previous section. Below, we will go through each task and describe the results according to what the evaluator observed during the test, as well as what we learned from analyzing the audio- and screen-output recordings. The results from the semi-structured interview conducted after the tasks had been accomplished, are also presented here. Note that all the quotes here have been translated from Norwegian.

#### Task 1

- **Without leaving the Wordpress Administration Screen, search for the plugin called *Name Redactor*. Once you have found the plugin, you will have the option of reading the documentation, but this is entirely up to you.**

There are basically three ways to install a plugin on a WordPress website. Because our solution had been uploaded to the WordPress Plugin Directory[12], this task assumed that the participant would use the easiest method, which is to download and install the plugin through the Administration panel in WordPress. By going to the Plugins screen and clicking on *Add New*, the user can either type in the name of the plugin in the search field, or search for it using any of the associated tags (e.g. *redact*). Clicking *Details* will display the *readme* file for the plugin, while clicking *Install Now* will install the plugin. Had the plugin not been hosted in the Wordpress Plugin Directory, the user would have had to download the plugin to their local computer and then either upload and install it using the built-in plugin installer in WordPress, or extract and then upload the plugin to the *plugins* folder in the WordPress directory on the server, using an FTP client software.

[12] `http://wordpress.org/extend/plugins/name-redactor/`

When it came to the task of searching for and finding the plugin, all the participants managed to do it within a reasonable amount of time. Everyone seemed to find it intuitive that in order to install a new plugin, they had to go to the Plugins menu and click on the *Add New* button.

A minor change was made to this task during the course of the study. For the first two participants, actually reading the documentation was part of the first task before proceeding with the second task. This was made optional for the remaining participants. The reason for this change was to better reflect a real-life scenario, where users may not be inclined to read through the whole documentation before starting using the plugin. We wanted to see if the users who decided not to read the documentation would still manage to use the plugin, or if not, at what point they would run into problems. We also wanted to see if the participants, in the event of becoming stuck with any of the tasks, would then read the documentation and whether or not they could easily find the information they needed in order to continue with the tasks.

Both of the participants who had to read the whole documentation as part of this task complained about the amount of text, saying that there was a bit too much to read, especially if they had to remember it all. As one participant remarked during the interview:

> "I am not sure if the average user would bother to read through the whole documentation, at least, for me personally, I am not sure that I would have bothered to do so."

When the same participant was asked about what they thought of the actual content in the documentation, they said that:

> "The explanation in the documentation was actually pretty decent, and you did not have to read the whole thing in order to understand what to do. But in order to make it even easier, the functionality of the plugin should be stated more precisely using less text, so that you do not have to spend more than ten seconds to understand what the program actually does."

After the task of reading the documentation was made optional for the remaining three participants, none of the participants chose to read it, instead reasoning that they could always come back to it later if they needed to. Two of the participants mentioned that they usually did not bother to read any documentation before using a program. One participant said that, in general, any more than one sentence of documentation would not be read unless they found themselves completely stuck.

**Task 2**

- **Again without leaving the Wordpress Administration Screen, install and activate the plugin.**

As mentioned above, after a plugin has been selected from the search result, the user needs to first download and install the application, by clicking on the *Install* button, and then activate it by clicking on the *Activate* button. Note that this is a design aspect of the WordPress platform itself, and therefore not something that our plugin has any influence over.

While all of the participants managed to install the plugin, one participant forgot to activate the plugin afterwards. Thinking that the task had been accomplished, the participant moved on to the next task.

**Task 3**

- **Write a blog post that contains a personal name. Tag the name for redaction using the plugin, and publish the post.**

Posts are entries, containing some form of content, that are usually displayed in reverse chronological order on the homepage of a WordPress site. To write a post, the user can either click on *Posts* and then *Add New* in the navigation menu on the left, or hover the mouse over the *+New* button at the top of the screen and select *Post*.

When writing a blog post, the user has the option of either using the *Visual* or *Text* mode of the editor. The Visual mode lets the user see the post as it will look like when it is published, while the Text mode shows the code, such as `<strong></strong>` for strong emphasis of text. The Text mode also replaces the *WYSIWYG* (What You See Is What You Get) editor buttons with so called *quicktags*. These buttons allows the user to quickly insert HTML tags into the content, either by clicking a button for inserting an opening tag, write some text, and then clicking the same button to insert a closing tag, or by selecting a portion of the text and then clicking a button for enclosing the text with the selected HTML tag. Our solution only displays a redact button in the Text editor, and not in the Visual editor.

While four of the participants were taken directly to the Text editor when clicking on *Post* followed by *Add New*, one participant was taken to the Visual editor, most likely because WordPress remembers which editor was used last, and automatically goes to the last used editor (all the participants used the same user account in the test). Because the participant had no previous experience using WordPress, and was not aware of the fact that there were two different editor modes, it resulted in the participant spending a couple of minutes looking for a button they could use for tagging a name. The participant was eventually able to find the Text editor, and immediately spotted the redact button there. Afterwards, the participant remarked that:

> "It would have been even better if both editor modes contained a redact button, instead of just the one."

Two of the participants who were taken directly to the Text editor, mistook the built-in WordPress tagging function used for grouping similar con-

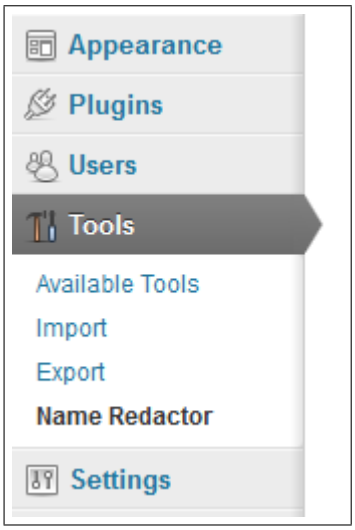


Figure 7.10: The Name Redactor settings menu, located under *Tools* in the navigation menu on the left.

tent. This function is located in the right hand menu when creating new content. So instead of tagging a name for redaction, they instead filed the post under the tagged name. One participant simply failed to spot the redact button, while in the case of the participant who had forgotten to activate the plugin, there was no redact button to use for tagging a personal name. Instead, both participants noticed the built-in WordPress tagging function and used that instead.

Of the participants who correctly managed to tag a name for redaction, one participant remarked that the button was a bit difficult to spot.

**Task 4**

- **Change the plugin settings, so that the program automatically tries to identify and tag personal names without you having to do it manually.**

This was perhaps the task where the participants encountered the most problems, as they had difficulty finding where the plugin settings menu was located. The Name Redactor plugin has its own settings menu, located in the navigation menu on the left, under the *Tools* top-level menu. (see Figure 7.10). As mentioned previously, one of the participants had forgotten to activate the plugin. This meant that the participant would have no possible chance of finding the plugin settings menu. In this case, the researcher subtly hinted that maybe they had missed something in a previous task. The participant then re-read the previous tasks, realised where they had gone wrong, and was able to correct their mistake.

Two of the participants reasoned that the plugin settings menu had to be located near or "in" the plugin itself, and therefore started to look for it in the *Installed Plugins Screen*, which is where users can see which plugins are installed and choose which plugins to activate or deactivate. From there they clicked on the plugin editor, which is where the user can modify the source code of a plugin. Realising that the they had ended up in the wrong place, they eventually managed to locate the correct menu, although they had to look through almost all the different menus in the WordPress Administration panel in order to find it. Four out the five participants also reasoned that the plugin settings menu should be located under *Settings* in the navigation menu. As one participant remarked after discovering the location of the plugin settings menu:

> "There is nothing that would suggest to me that the plugin settings menu is located under Tools."

Only one participant found the plugin settings menu almost immediately. When later asked what made the participant look there instead of e.g. the Wordpress Settings menu, the participant replied that they were somewhat familiar with how Wordpress functioned, and thought it was natural to find it there, because they thought of the plugin as a tool of sorts.

Once the participants managed to find the plugin settings menu, they were able to change the *redact mode*, although some of them had to spend some time reading the instructions located above the settings.

Below the *redact mode* settings, the user can set the plugin to check up against the opt-in/opt-out list. Although the next three tasks involved interacting with the opt-in/opt-out list, no task involved actually ticking off any of these checkboxes. However, two of the participants reasoned that these checkboxes had to be ticked off in order for the opt-in/opt-out list to function, and proceeded to do so.

Two participants mentioned that they missed some sort of feedback when clicking on the *Save Changes* button. When clicking on the button, there is nothing that informs the user that the settings have in fact been changed, other than that the screen jumps to the top of the page. While this sort of feedback was implemented for the other two settings tabs, the *Options* tab was unfortunately overlooked.

One participant remarked that there was too much text to read in order to understand how to do something.

**Task 5, 6 and 7**

- **Add a name to the list of names that should *always* be tagged automatically.**
- **Add a name to the list of names that should *not* be tagged automatically.**

- **Remove one of the names you just added.**

Because these tasks were so closely connected, they are grouped together in this presentation for easier reading.

While all the participants managed to locate the *Opt-in/opt-out* tab quickly, three participants remarked that they did not know what the terms opt-in and opt-out meant, and four participants expressed confusion regarding which list to add a name to in order for it to be tagged automatically. When later asked about the problem they had of adding names to the opt-in/opt-out lists, one participant explained that:

> "I was wondering what opt-in and opt-out means. The name of the tab was not obvious to me what it meant."

However, after reading the instructions located above the list, all of the participants managed to figure out which list to add a name to. One participant remarked that:

> "For me, the terms opt-in and opt-out does not mean anything. But it was very easy to understand once I read about it in the documentation."

Another participant remarked that it was very easy to get an overview of which names belonged to which list, and also liked the fact that they could sort the list according to opt-in/opt-out status.

One participant tried to add the same name to both lists, and received an error message saying that the name had already been added. The participant then navigated back to the *Options* tab, ticked off the checkbox telling the plugin to use the opt-out list (having previously already ticked off the other checkbox regarding the opt-in list), clicked save, and navigated back to the opt-in/opt-out tab. This time, the participant tried to add a different name, and succeeded.

After adding a name to each of the lists, all of the participants managed to delete one of the names without any difficulties.

#### Task 8

- **Deactivate and remove the plugin from WordPress.**

All the participants managed to complete this task without any difficulties, although one participant used the *Delete* option in the *Bulk Actions* dropdown menu instead of the *Delete* button next to the plugin.

### 7.2.4 Validity Evaluation

All of the participants were familiar to the evaluator, and it cannot be ruled out that this familiarity may have had an impact on the answers given during the interview.

Four out of the five participants had a job involving working with computers, and may therefore have an advantage when it comes to using the plugin, compared to people who are not so familiar with the use of computers. This may have had an impact on the test results.

### 7.2.5 Discussion

The aim of the usability test was to find out whether or not the solution was usable by the intended user group. Considering that the solution has been implemented as a plugin for the publishing platform WordPress, the intended user group could be defined as persons who writes a blog (also called bloggers). Unfortunately, only two of the five participants could be said to fall into this category, as both had their own WordPress blogs. The remaining three participants had never before written a blog. However, new and inexperienced WordPress users might also want to use our solution. And judging by the results from the usability test, the two participants who had prior experience writing a blog and using WordPress seemed to encounter many of the same problems as those who had no prior experience.

An important criteria that needs to be met in order for people to go to the trouble of actually using the application, is that it should be relatively easy to use. The findings from the usability study indicates that there are some problematic aspects that probably needs to be redesigned in the next version of the application. Some of these will be discussed below.

First of all, some of the terms used within the application were unfamiliar to many of the users. For example, three of the participants remarked that they did not understand the meaning of the terms *opt-in* and *opt-out*, and suggested that they should be called something else to make their functionality or purpose more intuitive.

The fact that the redact button only existed in one of the editors seemed to cause some confusion. When clicking on the *New Post* button to write a new blog post, one participant was taken to the Visual editor instead of the Text editor. After spending some time there looking for the redact button, they eventually found it in the Text editor. Another participant remarked that they usually only used the Visual editor when writing a blog post. This suggests that the Visual editor should probably also contain a redact button, and not just the Text editor. The main reason why it was not implemented in both editors is that adding a button to the Visual editors requires a lot more additional coding to achieve, and with a limited time schedule it was decided that the extra amount of time it would take would not be worth it.

However, this is something that could be implemented in a future version when time permits it.

At the time of developing the solution, a design decision was taken, which involved placing the settings screen for our plugin below the top-level menu *Tools* (as a sub-level menu). The reasoning behind this decision was that the plugin could be considered a tool for enhancing the privacy aspect of WordPress. However, in hindsight, this reasoning was likely wrong. After consulting the guide for determining the correct location for a sub-level menu item, provided in [61], the more correct choice would most likely have been to place the settings screen below the top-level menu *Settings* instead. As demonstrated by several of the participants, this placement would be more intuitive as well. This is probably something that should be changed in the next version of the application.

During the interview, one participant pointed out that perhaps the redact mode should be set to automatically detect and redact personal names by default when the plugin is activated for the first time, instead of the redact mode being set to only redact manually tagged personal names. This approach, commonly known as *privacy by default*, suggests that the highest privacy level should be the default setting, so that the user does not have to do any additional changes to the settings in order to have the highest form of privacy. The reasoning that lay behind the decision of having the redact mode set to only redact manually tagged names as the default setting, was that the user should have control over how content would be redacted from the start. If the default setting had been to automatically detect and redact personal names, then every word the application would identify as a personal name would be redacted as soon as the plugin had been activated. It is also considered bad form to have a plugin change any of the functionality on a website from the moment it is activated, without giving the administrator the chance to prevent it.

Lastly, there is the issue of the documentation (i.e., the readme file) of the application, namely that the amount of text contained within it seemed to overwhelm many of the participants. The amount of text may have been one of the reasons why so few of the participants chose to read it, even after they would become stuck on a task. For the participants who did return to the readme file after becoming stuck, the amount of text also made it difficult for the participants to actually find what they were looking for. This indicates that the readme file in a future version of the application should contain less text with a more precise wording. However, we cannot assume that users will always read the documentation before starting to use the plugin, as this study indicates. After the activity of reading the documentation at the start of the study was no longer mandatory, none of the participants bothered to read it. And judging from the remarks of some of the participants, they probably would not bother to read it even though the amount of text had been lessened. One way to address this problem could be to attempt to make

the plugin interface more intuitive (e.g., the plugin settings menu should be located where most people expect it to be), as well as make the names of the functions easier to understand without having to look them up in the documentation.

## 7.3 Summary

In this chapter we have evaluated the solution in terms of functionality, by verifying that it functioned as intended, and in terms of usability, by having a group of people interact with it and perform a set of basic tasks. In the following chapter we summarize and conclude this thesis.

# Chapter 8

# Conclusion and Future Work

At the beginning of this thesis, we specified a set of research objectives that we identified as important in order to achieve the overall goal, which was to develop a tool for facilitating increased control over personal data published online. This chapter summarizes the results of this work, and presents some reflections on possible future work.

## 8.1 Legal Background

As part of the work done for this thesis, we clarified the concepts of privacy and data protection. From this, we learned that the issue of data protection is very much a case of whether or not information can be linked to an identified or identifiable individual. This has served as motivation for the development of a solution for shielding personal names from search engines.

## 8.2 Robot Directives

As part of our research into how publishers can protect their personal data from search engines, we conducted an in-depth study on the *Robots Exclusion Protocol* (REP). From that study, we found that the REP seems to be quite reliable as a method for discouraging search engines from crawling and indexing content on a directory-level or page-level. However, the lack of an official standards body or RFC for the REP makes it difficult to find information regarding its implementation. The fact that different search engines support different directives, or may interpret the directives differently, adds to the difficulty of using it.

Among our findings from the study, there were a few unexpected results which may warrant a future study. Unfortunately, due to limited time, we were unable to refine the experiment and perform additional tests. Our study only included the Google search engine. For a future study on the REP, it would be interesting to include other search engines as well. While our

study indicates that Google could be considered quite good at adhering to the directives, we do not know if this adherence is shared by other search engines. Such a test would ideally include not just the ones who officially state their support of the REP, but also the ones who makes no such statement.

A related study [46, 47], mentioned in Chapter 4, revealed many incorrect usages of the REP. Hopefully, the study conducted as part of this thesis may contribute to a better understanding of how robot directives can be used to control how search engine robots crawls and indexes websites.

## 8.3 Name Redactor

As already mentioned, the main goal of this thesis was to develop a tool for facilitating increased control over personal data published online. Our solution, implemented as a plugin for the WordPress publishing platform, achieves this goal by allowing publishers to redact personal names from content that is viewed by search engine robots, with the aim of avoiding personal names being indexed along with any contextual data attached to them.

In addition to developing the plugin, we also managed to get it accepted into the WordPress Plugin Directory. Among other things, this makes it easier to maintain the project, as well as providing a greater exposure to the WordPress community.

The solution was evaluated in terms of functionality and usability. From the results of the software testing we were able to verify that it correctly redacted personal names from content that was viewed by a search engine robot. And while the algorithm for automatically identifying personal names was not completely accurate, it still managed to correctly identify all personal names included in the content. Future work on the plugin could include adding more advanced rules for name detection. However, this is a very complicated area in the field of computer linguistics, and creating a robust algorithm is probably a master thesis in its own right. Earlier attempts of adding more advanced rules to our plugin did not turn out to be very successful, and was eventually scrapped in favour of more simplistic rules that were easier to predict the outcome of. The results from the usability testing provided us with lots of useful feedback, including highlighting some problematic usability aspects, which should be taken into account in a possible future version.

Finally, while the solution presented in this thesis have been developed for the WordPress platform, the functionality of it should not be too difficult to port over to another publishing platform that features a similar plugin architecture.

# References


[1] "Vg nett: La ut hemmelig pasientinfo på nett - klarte ikke fjerne den." http://www.vg.no/helse/artikkel.php?artid=574012. Accessed 28 April 2013.

[2] "Digi.no: Enorm lekkasje i groupon." http://www.digi.no/872887/enorm-lekkasje-i-groupon. Accessed 28 April 2013.

[3] "Online reputation in a connected world." go.microsoft.com/?linkid=9709510. Accessed 04 May 2012.

[4] Personvernkommisjonen, "Nou 2009:1 - individ og integritet - personvern i det digitale samfunnet," tech. rep., Fornyings- og administrasjonsdepartementet, 2009.

[5] L. A. Bygrave, "Privacy and data protection in an international perspective," in *56 ICT Legal Issues, October 2010*, vol. 56, pp. 165–200, Stockholm Institute for Scandinavian Law, 2010.

[6] J. DeCew, "Privacy," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), The Metaphysics Research Lab Center for the Study of Language and Information Stanford University Stanford, CA 94305-4115, fall 2008 ed., 2008.

[7] S. D. Warren and L. D. Brandeis, "The right to privacy," *Harvard Law Review*, vol. V. IV, No. 5, December 1890.

[8] "Mandat for personvernkommisjonen." http://www.regjeringen.no/upload/FAD/Vedlegg/Statsforvaltning/Mandat_Personvernkommisjonen.pdf. Accessed 05 June 2012.

[9] "The universal declaration of human rights," 1948.

[10] "Convention for the protection of human rights and fundamental freedoms," 1950.

[11] "Data protection." http://www.coe.int/t/dghl/standardsetting/DataProtection/default_en.asp. Accessed 03 June 2012.

[12] "Convention for the protection of individuals with regard to automatic processing of personal data," 1981.

[13] "Personal data act (english version)." http://www.datatilsynet.no/Global/english/Personal_Data_Act_20120420.pdf, April 2000. Accessed 03 May 2013.

[14] "Data inspectorate's homepage." http://datatilsynet.no/English/. Accessed 03 June 2012.

[15] "Personally identifiable information." http://en.wikipedia.org/wiki/Personally_identifiable_information. Accessed 13 May 2012.

[16] "Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," October 1995.

[17] "Judgment of the court of 6 november 2003 - criminal proceedings against bodil lindqvist." http://curia.europa.eu/juris/liste.jsf?language=en&num=C-101/01, Nov 2003.

[18] "Privacy-enhancing technologies: The path to anonymity (volume 1)," Aug 1995.

[19] "Privacy-enhancing technologies: The path to anonymity (volume 2)," Aug 1995.

[20] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, pp. 84–90, Feb. 1981.

[21] Y. Deswarte and C. Aguilar Melchor, "Current and future privacy enhancing technologies for the internet," *Annales Des Télécommunications*, vol. 61, pp. 399–417, 2006.

[22] "Communication from the commission to the european parliament and the council on promoting data protection by privacy enhancing technologies (pets)," 2007.

[23] H. Burkert, "Privacy-enhancing technologies: typology, critique, vision," in *Technology and privacy* (P. E. Agre and M. Rotenberg, eds.), pp. 125–142, Cambridge, MA, USA: MIT Press, 1997.

[24] "Data protection guidance note: Privacy enhancing technologies (pets)." http://www.ico.gov.uk/upload/documents/library/data_protection/detailed_specialist_guides/privacy_enhancing_technologies_v2.pdf. Accessed 4 May 2012.

[25] G. van Blarkom, J. Borking, and J. Olk, *Handbook of Privacy and Privacy-Enhancing Technologies (The Case of Intelligent Software Agents)*. The Hague, The Netherlands: College bescherming persoonsgegevens, 2003. (Draft version).

[26] "St.meld. nr. 17 (2006-2007) eit informasjonssamfunn for alle," 2006.

[27] "First report on the implementation of the data protection directive (95/46/ec)," 2003.

[28] A. Cavoukian, "Privacy by design - the 7 foundational principles." `http://www.privacybydesign.ca/content/uploads/2009/08/7foundationalprinciples.pdf`, August 2009. Accessed 9 Marts 2012.

[29] H. Tavani, "Search engines and ethics," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), The Metaphysics Research Lab Center for the Study of Language and Information Stanford University Stanford, CA 94305-4115, fall 2012 ed., 2012.

[30] "Hypertext transfer protocol – http/1.1." `http://tools.ietf.org/html/rfc2616#section-14.43`. Accessed 18 May 2012.

[31] "Search engine market share." `http://marketshare.hitslink.com/search-engine-market-share.aspx?qprid=4`. Accessed 9 May 2012.

[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[33] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons Ltd, 2007.

[34] A. H. Jørgensen, "Thinking-aloud in user interface design: a method promoting cognitive ergonomics," *Ergonomics*, vol. 33, no. 4, pp. 501–507, 1990.

[35] T. Boren and J. Ramey, "Thinking aloud: reconciling theory and practice," *Professional Communication, IEEE Transactions on*, vol. 43, no. 3, pp. 261–278, 2000.

[36] "A standard for robot exclusion." `http://www.robotstxt.org/orig.html`. Accessed 10 May 2012.

[37] "About /robots.txt." `http://www.robotstxt.org/robotstxt.html`. Accessed 10 May 2012.

[38] "Prevent a bot from getting "lost in space" (sem 101)." `http://www.bing.com/community/site_blogs/b/webmaster/archive/2009/08/`

21/prevent-a-bot-from-getting-lost-in-space-sem-101.aspx, August 2009. Accessed 11 May 2012.

[39] "Robots.txt specifications." https://developers.google.com/webmasters/control-crawl-index/docs/robots_txt. Accessed 29 April 2013.

[40] "How to create a robots.txt file." http://www.bing.com/webmaster/help/how-to-create-a-robots-txt-file-cb7c31ec. Accessed 29 April 2013.

[41] "Controlling crawling and indexing: Getting started." https://developers.google.com/webmasters/control-crawl-index/docs/getting_started?hl=no. Accessed 05 October 2012.

[42] "Robots meta tag and x-robots-tag http header specifications." https://developers.google.com/webmasters/control-crawl-index/docs/robots_meta_tag. Accessed 11 May 2012.

[43] "Robots and the meta element." http://www.w3.org/TR/html401/appendix/notes.html#h-B.4.1.2. Accessed 29 April 2013.

[44] "Dmoz open directory project." http://www.dmoz.org/. Accessed 11 May 2012.

[45] "Standard for interchange of usenet messages (url)." http://www.ietf.org/rfc/rfc0850.txt, June 1983. Accessed 15 October 2012.

[46] Y. Sun, Z. Zhuang, and C. L. Giles, "A large-scale study of robots.txt," in *Proceedings of the 16th international conference on World Wide Web*, WWW '07, (New York, NY, USA), pp. 1123–1124, ACM, 2007.

[47] Y. Sun, Z. Zhuang, I. G. Councill, and C. L. Giles, "Determining bias to search engines from robots.txt," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, (Washington, DC, USA), pp. 149–155, IEEE Computer Society, 2007.

[48] S. Kolay, P. D'Alberto, A. Dasdan, and A. Bhattacharjee, "A larger scale study of robots.txt," in *Proceedings of the 17th international conference on World Wide Web*, WWW '08, (New York, NY, USA), pp. 1171–1172, ACM, 2008.

[49] "Duplicate content." http://support.google.com/webmasters/bin/answer.py?hl=en&answer=66359. Accessed 29 April 2013.

[50] "Microsoft and yahoo seal web deal." http://news.bbc.co.uk/2/hi/business/8174763.stm. Accessed 13 January 2013.

[51] "Why is my site not in the index?." http://www.bing.com/webmaster/help/why-is-my-site-not-in-the-index-2141dfab. Accessed 03 January 2013.

[52] "Operators and more search help." http://support.google.com/websearch/bin/answer.py?hl=en&answer=136861. Accessed 11 January 2013.

[53] "Advanced search keywords." http://onlinehelp.microsoft.com/en-us/bing/ff808421.aspx. Accessed 11 January 2013.

[54] "About sitemaps." http://support.google.com/webmasters/bin/answer.py?hl=en&answer=156184. Accessed 29 April 2013.

[55] "About wordpress." http://wordpress.org/about/. Accessed 22 Marts 2012.

[56] "Usage of content management systems for websites." http://w3techs.com/technologies/overview/content_management/all. Accessed 30 Marts 2012.

[57] "Settings reading screen." http://codex.wordpress.org/Settings_Reading_Screen. Accessed 29 April 2013.

[58] "Managing plugins." http://codex.wordpress.org/Managing_Plugins. Accessed 8 April 2013.

[59] "Merriam webster dictionary: Redact." http://www.merriam-webster.com/dictionary/redact. Accessed 22 April 2013.

[60] "Merriam webster dictionary: Redactor." http://www.merriam-webster.com/dictionary/redactor. Accessed 22 April 2013.

[61] "Administration menus." http://codex.wordpress.org/Adding_Administration_Menus. Accessed 29 April 2013.

[62] "Writing a plugin." http://codex.wordpress.org/Writing_a_Plugin. Accessed 29 April 2013.

[63] "Google crawlers." http://support.google.com/webmasters/bin/answer.py?hl=en&answer=1061943. Accessed 15 April 2013.

[64] "Wordpress can host your plugin." http://wordpress.org/extend/plugins/about/. Accessed 22 April 2013.

[65] "Obligation to notify?." http://www.nsd.uib.no/personvern/en/notification_duty/. Accessed 10 April 2013.

# Appendix A

# How to Obtain the Software

The software for the solution presented in this thesis is being hosted by WordPress, and can be found in the WordPress Plugin Directory, at the following address:

`http://wordpress.org/extend/plugins/name-redactor/`